July 2016

Randomized Computations for Efficient and Robust Finite Element Domain Decomposition Methods in Electromagnetics

Wei Wang

RANDOMIZED COMPUTATIONS FOR EFFICIENT AND ROBUST FINITE ELEMENT DOMAIN DECOMPOSITION METHODS IN ELECTROMAGNETICS

A Dissertation Presented

by

WEI WANG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2016

Electrical & Computer Engineering

RANDOMIZED COMPUTATIONS FOR EFFICIENT AND ROBUST FINITE ELEMENT DOMAIN DECOMPOSITION METHODS IN ELECTROMAGNETICS

A Dissertation Presented

by

WEI WANG

Approved as to style and content by:

Marinos Vouvakis, Chair

Christopher Hollot, Member

Eric Polizzi, Member

Blair Perot, Member

Christopher Hollot, Department Chair
Electrical & Computer Engineering

DEDICATION

To my beloved parents and wife.

ACKNOWLEDGMENTS

I have to start my acknowledgements with my advisor Prof. Marinos N. Vouvakis. I am so grateful to him for his technical advice, encouragement, and support during my graduate study. I am particularly grateful to his challenging questions and training to make me think out of the box.

I would like to thank the members of the Antenna Propagation Laboratory at University of Massachusetts Amherst especially Georgios Paraschos who built a solid foundation of the domain decomposition codes. I would like to extend my thanks to Simon Tang, Michael Lee, John Logan, Javad Moshfegh for their help. The thanks also extends to Dimitrios Makris and Zekios Constantinos who helped proofreading this manuscript.

Furthermore, I would like to thank my undergraduate thesis advisor, Prof. Zhong-Qing Zhang at Zhejiang University, China, for his support and introduced me into the computational electromagnetics area.

Thanks are extended to my friends especially Kan Fu, Zuo-Jing Chen and Yan-bo Wang for their friendship in Amherst.

Last, but certainly not the least, I want to thank my parents, Jiapei Wang and Hongyu Wei. I am so grateful to you for your constant support, patience, and love. Very special thanks should be extended to my wife, Sha Yu, for her encouragement during my difficult times.

ABSTRACT

RANDOMIZED COMPUTATIONS FOR EFFICIENT AND ROBUST FINITE ELEMENT DOMAIN DECOMPOSITION METHODS IN ELECTROMAGNETICS

MAY 2016

WEI WANG

B.Sc., ZHEJIANG UNIVERSITY, HANGZHOU, CHINA, JUNE 2007

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST, FEBRUARY 2010

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Marinos Vouvakis

Numerical modeling of electromagnetic (EM) phenomenon has proved to become an effective and efficient tool in design and optimization of modern electronic devices, integrated circuits (IC) and RF systems. However the generality, efficiency and reliability/resilience of the computational EM solver is often criticised due to the fact that the underlying characteristics of the simulated problems are usually different, which makes the development of a general, "black-box" EM solver to be a difficult task.

In this work, we aim to propose a reliable/resilient, scalable and efficient finite elements based domain decomposition method (FE-DDM) as a general CEM solver to tackle such ultimate CEM problems to some extent. We recognize the rank deficiency property of the Dirichlet-to-Neumann (DtN) operators involved in the previously proposed FETI-2λ DDM formulation and apply such principle to improve the compu-

tational efficiency and robustness of FETI-2λ DDM. Specifically, the rank deficient DtN operator is computed by a randomized computation method that was originally proposed to approximate matrix singular value decomposition (SVD). Numerical results show a up to 35% run-time and 75% memory saving of the DtN operators computation can be achieved on a realistic example. Later, such rank deficiency principle is incorporated into a new global DDM preconditioner (W-FETI) that is inspired by the matrix Woodbury identity. Numerical study of the eigenspectrum shows the validity of the proposed W-FETI global preconditioner. Several industrial-scaled examples show significant iterative convergence advantage of W-FETI that uses 35%-80% matrix-vector-products (MxVs) than state-of-the-art DDM solvers.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

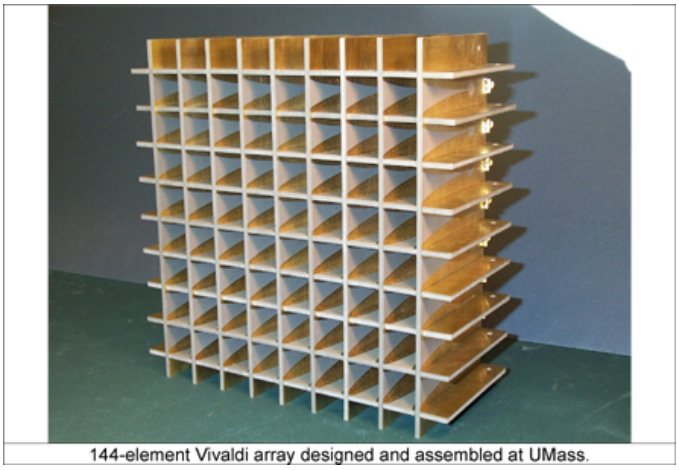
INTRODUCTION

1.1 Problem Statement

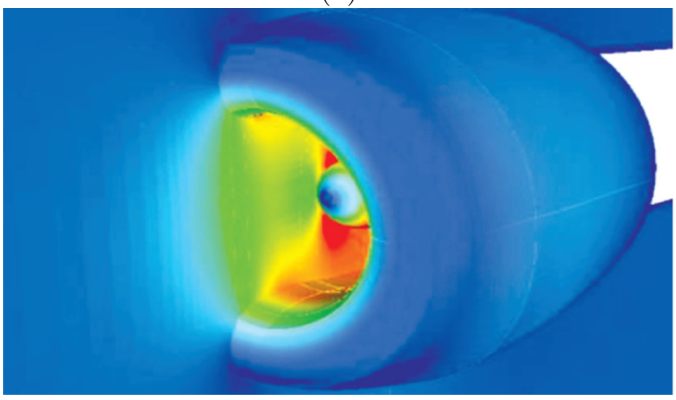
Modern engineering design of radio frequency, microwave, milimiterwave and optical devices and systems, as shown in Fig. 1.1(a)-(c), almost exclusively rely on computational electromagnetic (CEM) models and tools as well as theoretical intuition. The main challenge in such computations lies in devising methods and algorithms that *efficiently* and *reliably/resiliently* [1] solve Maxwell's equations in arbitrary structures and excitation scenarioes.

As revealed in Fig. 1.2, the clock speed of modern CPU has levelled off in recent years, while the number of cores per CPU/processor keeps growing. Following this trend, in the modern area of multi/many-core and distributed memory computing platforms, CEM methods and algorithms must leverage computing parallelism [3] and out-of-core computations [4] for a chance at coping with the increasing complexity of the systems and devices that are called to simulate.

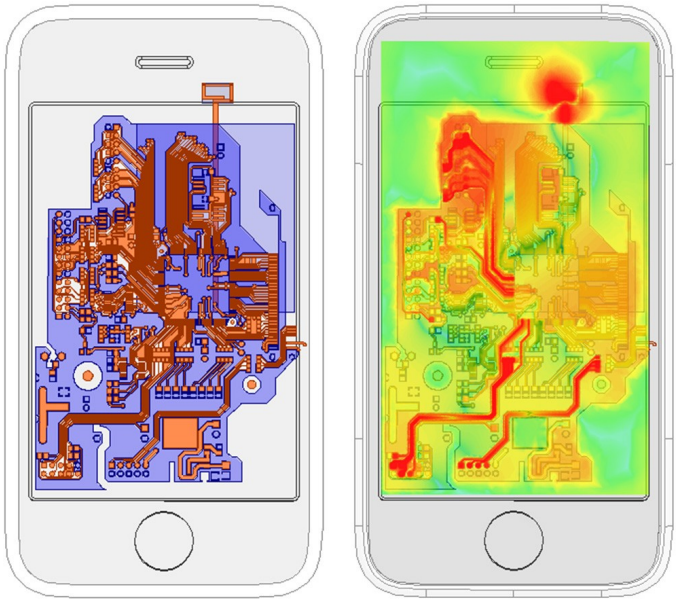
In a recent resurgence the broad field of algorithmic computing, randomized algorithms [5–7] have shown promising signs in making conventional algorithms more reliable/resilient [8], efficient [9] and suitable to parallel and out-of-core computations [10]. The main research questions that will be posed in this work are: can randomized algorithms, more specifically linear algebra algorithms, be used to improve the efficiency, reliability, and parallel and out-of-core efficiency of CEM methods and algorithms? and if so, how? Our hypothesis is that if these randomized linear algebra algorithms are appropriately combined with deterministic CEM methods such as



(a)



(b)



(c)

Figure 1.1. Examples of challenging computational problems; (a) Wideband Vivaldi antenna array (source: UMass Amherst); (b) A commerical Airbus jet intake simulation (source: MSC software); (c) Radiated EMI simulation from mobile devices (source: ansys.com).

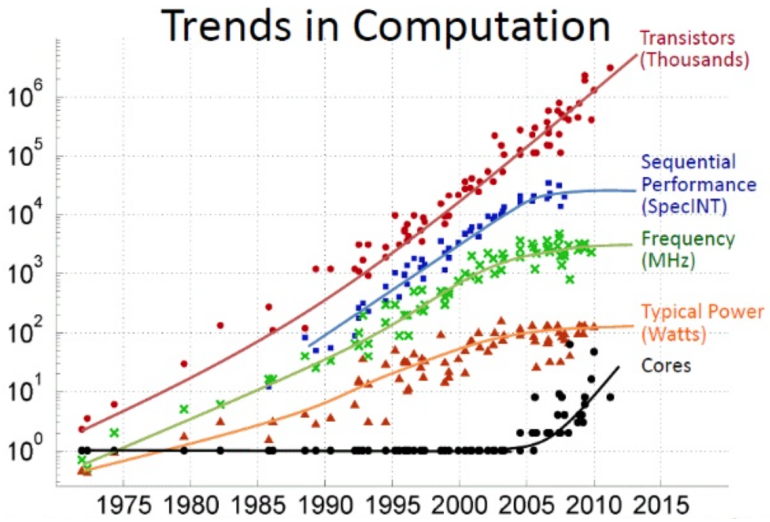


Figure 1.2. The evolution of CPU technology. Core clock speed keeps levelling off in recent years (source: ANSYS, Advances in accelerator-based CFD simulation).

domain decomposition methods (DDM) and the finite element methods (FEM), they can provide significant improvements over the state-of-the-art.

1.2 Importance

The proposed domain decomposition finite element method is a general full-wave EM solver that can be used in any electronic or photonic problems requiring solution of Maxwell's equations. Full-wave modeling has become the industrial standard of designing high-frequency electronic systems such as phased arrays and RF devices. Although successful, it usually comes at a premium computational overhead in both time and memory for complex and large-scale problems. For instances, the MG-FETI DD method [1], considered to be the state-of-the-art DDM, needs more than 12 hours and 13 GB RAM [1] to perform a full-wave simulation of a moderate size printed circuit board as shown in Fig. 1.3 on a single workstation for one frequency. Such computational burden becomes more serious when one is interested in system wideband performance. Meanwhile, the multi-level fast multipole algorithm (MLFMA) [11] was proposed and developed for solving electrically large EM problems

as it is based on the plane wave expansion, however without elaborate modification, it performs considerably poor at low frequency problems [12]. Another example is multigrid methods [13], they perform superbly at electrically small or low-frequency problems, but scale unfavorably with electrical size or frequency.

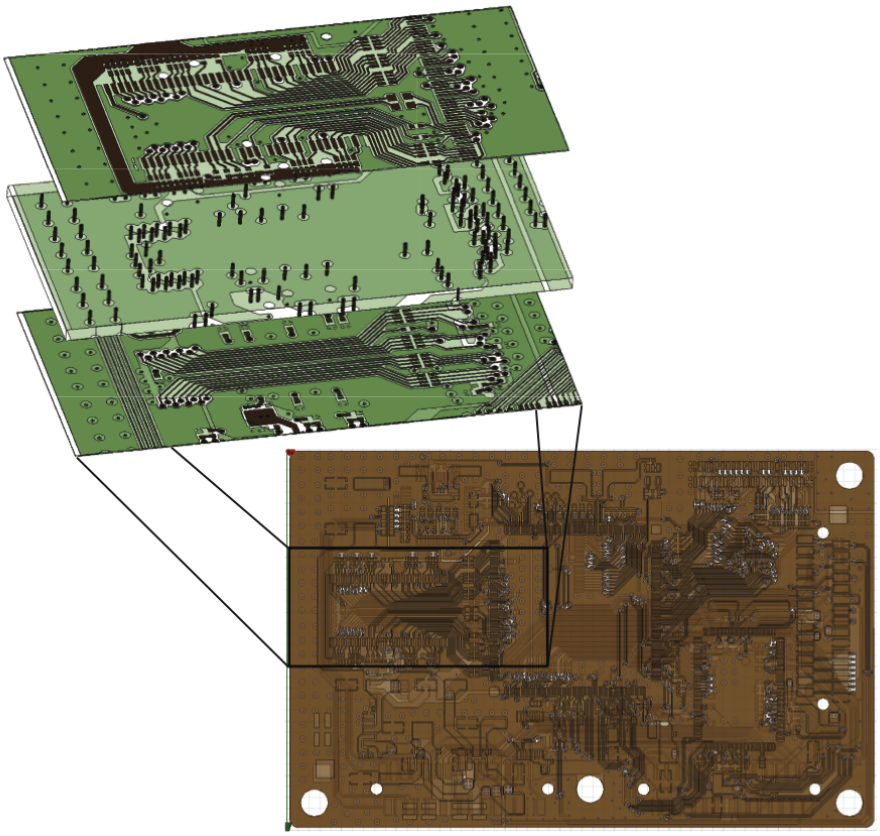


Figure 1.3. A complex multi-layer printed circuit board model. The illustrative example was simulated on a workstation with 2 Intel Xeon 3.2 GHz quad-core processors.

Overall, the success of the current-generation computational EM method appears to be limited for one or several types of problems and generally demands high computational resources. This dissertation work attempts to cure these problems in an alternative perspective. It will fully take advantage of the "magic" robustness and efficiency brought by the randomized linear algebra algorithms to develop a reliable/resilient and efficient full-wave domain decomposition EM solver. The proposed work will impact the next-generation RF and microwave engineering design

and optimization by fully leveraging the multi/many-core and distributed memory computational platforms. It is expected to significantly improve the computational efficiency of the Finite Element Tearing and Interconnecting (FETI)-2λ [1] domain discrete Dirichlet-to-Neumann map operators by at least 20%-50% in both run-time and memory. On the other hand, a tailored global preconditioning technique combining with randomized singular value decomposition [5] is estimated to greatly enhance the iterative convergence performance of the DDM compared to other state-of-art methods. The proposed robust and efficient DD framework will also be readily applied to problems with structured and unstructured decomposition as well as conforming and non-conforming meshing, leading to desired flexiblity in the pre-processing stage. It is also important to realize the proposed preconditoning approach is fully algebraic, meaning it is general and can be easily extended to other computational disiplines.

1.3 Literature Review

The literature review is divided into three parts. First an introduction of hybrid methods are presented, then a review of domain decomposition methods are given which covers various formulations of DD methods, advanced accelerators and precondtioning techniques and efficiency improvement including its parallel implementation. Finally a review of randomized linear algebra algorithms are provided which mainly focus on low-rank matrix approximation and its applications to fast singular value decomposition (SVD) computation.

1.3.1 Hybrid Methods

Hybrid methods are often appealing for analyzing unbounded EM radiation and scattering from heterogeneous structures [14]. The domain decomposition based finite element-boundary element coupling method [14, 15] provides an internal resonance free and symmetric DD formulation while reserving the mesh modularity between

FEM and BEM, and guarantees the spectral radius less or equals to one. Another example is the DDM in conjunction with integral equations [16], which can relieve the burden of mesh generation of complex objects while the non-overlapping DD provides a computationally efficient and effective preconditioner for usually ill-conditioned IE matrix. One major drawback of these hybrid methods is the $\mathcal{O}(N_s^2)$ computation complexity where N_s is the number of surface unknowns.

On the other hand, commerical simulation packages like Ansys HFSS [17] and FEKO [18] offer suites of different methods/solvers, i.e., multigrid finite element method (MG-FEM), MLFMA accelerated method of moment (MoM), physical optics (PO) or their hybrids. Inevitably, this increases the development complexity, but more importantly decreases user friendliness, as the decision on matching a problem to its best possible solver is one that requires significant tool training and experiences for the user.

1.3.2 Domain Decomposition Methods

1.3.2.1 DD Formulations

The DDM could date back to the original alternating Schwarz method [19] where the domain information are exchanged through interfaces with a Dirichlet Transmission Conditions [20]. The balancing domain decomposition (BDD) proposed by Mandel [21] was successful for scalr elliptic problems which has a bounded condition number of $(1+\log(H/h))^2$. However the method is not suitable for EM problems due to the *internal resonance* issue [22, 23]. The BDD with constraints (BDDC) [24] was proposed as a improved version of BDD. It uses an additional coarse problem that arises from the enforcement of constraints at certain DoFs to achieve better matrix conditioning. Unfortunately such approach is still not applicable for EM applications due to the internal resonance problem. The most attracting and successful method is probably the FETI (Finite Element Tearing and Interconnecting) method proposed

by Farhat and Roux [25]. It uses the conforming finite elements and Dirichlet TCs to enforce the field continuity with one set of Lagrange Multipliers (LMs). Since then, FETI method has been extended and improved in various ways. Wolfe [26] was the first to introduce FETI into CEM but the method suffered from internal resonance. FETI-"like", proposed by Vouvakis et al. [27], uses two sets of LMs on the domain interface and was the first method in FETI family that does not suffer from the internal resonance as it iterates on both the primal and dual variables, it is worth noting that FETI-"like" to some extent is a predecessor of the FETI-2λ method that is used in this work. More recently, Li et al. [28] developed the FETI-DP [29] variant - FETI-DPEM which is suitable for EM applications as it enforces the tangential continuity across the domain interfaces with the Dirichlet TCs. It improved version termed as FETI-DPEM2 was later proposed in [30] which adopts the impedance type TCs to improve the robustness. For various DDM formulations with further discussion, interested readers may refer to [20, 31, 32].

1.3.2.2 Accelerators and Preconditioning Techniques

In the past decade, a significant amount of research have been conducted to improve the validity and robustness of DDM for solving electromagnetic problems. Among those, probably the most common approach in a DD method is to recast the decomposted boundary value problem with suitable transimission conditions. TCs are critical in developing the interface equation that must be later solved iteratively, as they help to enforce some field continuity across domains. Early use of Dirichlet and Neumann TCs proved successful in definite elliptic partial differential equation (PDE) problems, but when used for indefinite PDE problems, it failed to produce convergent algorithms [33]. Lions in [33] was first to propose Robin type TCs that worked on certain cases of indefinite PDEs. Despres in [34] extended Robin TCs to Maxwell's equations to TCs that are analogous to the EM impedance boundary

conditions [23] for successful convergence of the non-overlapping DDM for radiation dominated problems, unfortunately the method can not guarantee success in evanescent field dominated problems. Recently, Rawat [35] proposed a second-order TCs (SOTCs) which could shift both problematic TE and TM evanescent eigenvalues away from the spectrum origin and showed superior convergence over conventional complex Robin-type TCs.

Another important way to improve the computational robustness of DDMs is applying preconditioners that includes local and global acceleration techniques. Local DD acceleration refers to mechanisms that enhance information exchange between neighboring domains at each iteration. These methods [36, 37] aim to improve the convergence rate without sacrificing much parallel scalability as information is communicated among neighboring domains that mostly are distributed within the same computer node/processor. In this work, a locally exact algebraic preconditioner (LEAP) proposed in [1] is used as local acceleration methods, as shown in Fig. 1.4 for a 1D decomposition example.

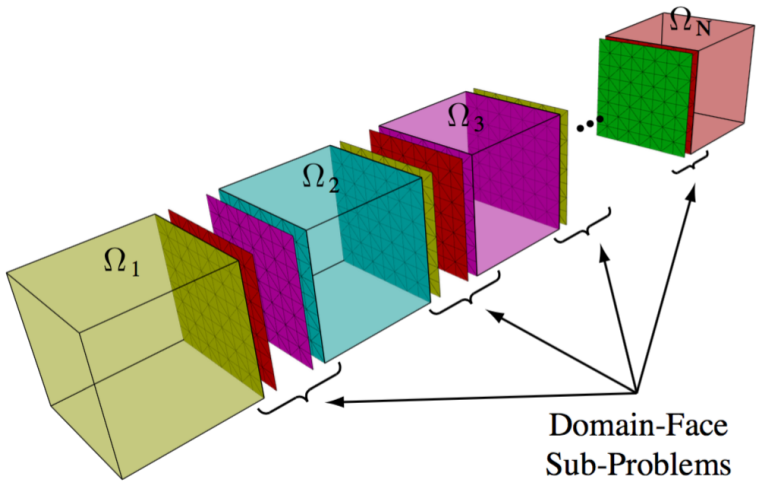


Figure 1.4. Local independent problems that constitute the LEAP preconditioner in a 1D decomposition [1].

Unfortunately a DDM with only local acceleration is not *scalable* w.r.t. domain count [20] as local preconditioners are not able to convey information throughout

domains at each iteration but only its neighboring ones. Global preconditioning is responsible for scalability w.r.t. domain count and electrical size of the computational problem, without an effective and efficient global preconditioning scheme, DD methods can not be reliably used to solve electrically large problems or even electrically small problems but decomposed into a considerable number of domains. To justify that, take the case of a simple Jacobi-based DD scheme, without any global preconditioning, in each DD matrix vector multiplication, the residual error only propagates to its neighboring domains. Therefore, per the definition of scalability defined in [20], this method would not be scalable as the number of iterations needed to propagate this residual error across the computational domain is proportional to the number of domains. A global preconditioner helps to convey information from one domain to all other domains at once at each matrix vector multiplication. A number of global preconditioning techniques have been proposed since the introduction of DDM into CEM, here we will conduct a brief review of these methods.

The wire-basket method proposed by Dryja et. al. [38] offers an efficient preconditioning for symmetric positive definite matrix and is designed in conjunction with primal DD methods, whereas this works employes a dual method (FETI-2λ). It involves a global problem with DoF residing on the collection of domain edges over the entire computational domain. The dual-primal FETI (FETI-DP) methods introduced by Farhat et. al. [29] combined the wire-basket idea with dual FETI methods. In all FETI-DP method and its variants, a small number of continuity constraints are enforced with primal unknowns as in wire-basket method across the domain interfaces in each matrix vector multiplication. The FETI-DP method contructs a coarse problem by eliminating the Lagarange Multipliers (LMs) or dual variables at the domain corner edges, and substituting them with primal variables. Indeed, the enforcement of the additional constraints in each iteration makes the local problems non-singular and at the same time provides an underlying coarse global problem [20]. One draw-

back of the FETI-DP method is the preconditioner size is dictated by the number of primal unknowns (fine grid edges and/or nodes) that reside along the domain edges, thus the size of the coarse problem is determined by the discretization level of the computational model. The performance of the preconditioner could degrade for a large size domains, and fine discretizations [1]. Toselli [39,40] was the first to develop a FETI-DP for edge finite elements for Maxwell's equation. It was shown that the condition number is bounded as $\mathcal{O}(1+\log(\frac{H}{h}))^4$ (H: model electrical size, h: mesh discretization) which only depends on the discretization level and the size of the domain for the computational model. Farhat et al. in [22] constructs an auxilary coarse problem by using plane wave basis functions as a global preconditioner for solving Helmolthz equation, it proves to be efficient to improve the DD scalability w.r.t. domain count on indefinite problems. Such concept was introduced by Peng et al. in [41] who applied the plane wave based preconditioner into the FETI-"like" DD scheme. More recently, a new global preconditioner termed as MG-FETI based on multigrid for the FETI-2λ formulation was proposed in [1]. It constructs a global coarse problem, shown in Fig. 1.5, that is capable of capturing the characteristic error modes by using a few macro-FE basis functions associated with each domain interface. Numerical results in [1] show that the combination of the LEAP local preconditioners and MG-FETI significantly outperforms FETI-DP on the convergence rate and run-time, while the MG-FETI coarse problem can still remain with a considerably small size. However, a potential issue with the MG-FETI global preconditioner lies in the ad-hoc nature of the coarse space basis function used. The MG-FETI uses basis functions that are not able to detect the physical characteristics of the field and this issue can be exaggerated for problems with complex structure. Table 1.1 summarizes some advantages and disadvantages of the forementioned preconditioning techniques.

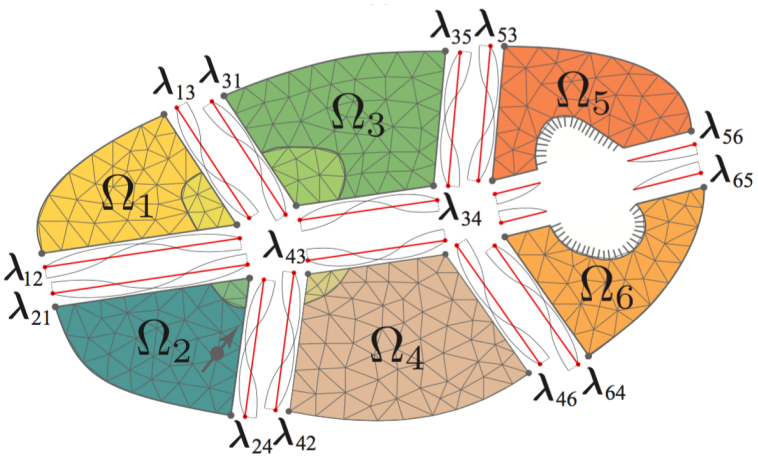


Figure 1.5. An auxilary multigrid problem with coarse LM discretization at domain interfaces [1].

Table 1.1. DDM Preconditioning in CEM

	Method	Advantages	Disadvantages
LOCAL	Impedance TC [42]	Simple and works well on radiation problems	Poor iterative convergence
	SOTC [35]	Scalable wrt h	Problem dependent parameter tuning
	LEAP [1]	Algebraic, scalable wrt h & domain size	Sensitive to buffer size
GLOBAL	FETI-DP [29]	Avoids domain-edge singularities	Large sparse global preconditioning matrix
	Planewave [41]	Scalable wrt domain size	Ill-conditioned preconditioning matrix with increasing kD
	MG-FETI [1]	Scalable wrt domain size, small global matrix	Ad-hoc chosen basis functions

1.3.2.3 Efficiency Improvement

The computational efficiency of DDM remains as another challenging issue. It is obvious that DDM is developed meant for large-scale simulation by leveraging parallel computing architectures, the computational efficiency of DDM can be affected by two primary ways:

(1) Load balancing and communication overhead. Load balancing is closely related to domain partition, which has been proven to be a NP-complete problem [43]. Therefore it is very time consuming to find an optimal domain partitions. In the past decades, various approaches have been proposed [3] to solve this problem. In this work, we mainly adopt graph partitioning package METIS [44] to decompose the original computational domain due to its high quality of generated subdomains. Once a domain partition is achieved, load balancing is about to assign domains to different processors that minimize the load difference between each processor. Then, the communication

overhead is mainly related to the neighboring communication and this can be reduced by assigning neighboring domains in the same computing node and minimizing the domain interface mesh size. It is important to note that better load balance and minimized communication overhead are usually exclusive to each other. To achieve better load balance, it is preferred to decompose the original problem with many domains with considerably smaller size, however this strategy usually increases the number of domains, therefore the resources required by the interface problem increases, eventually leading to more communication overhead. Fig. 1.6 show a comparison of two different domain partition for the same printed circuit board model. The partition with 15 domains generates 22 interfaces while the other generates in a total of 136 interfaces. It was suggested in [45] that the domain DoFs between 7,000-25,000 generally gives a considerably efficient DDM implementation.

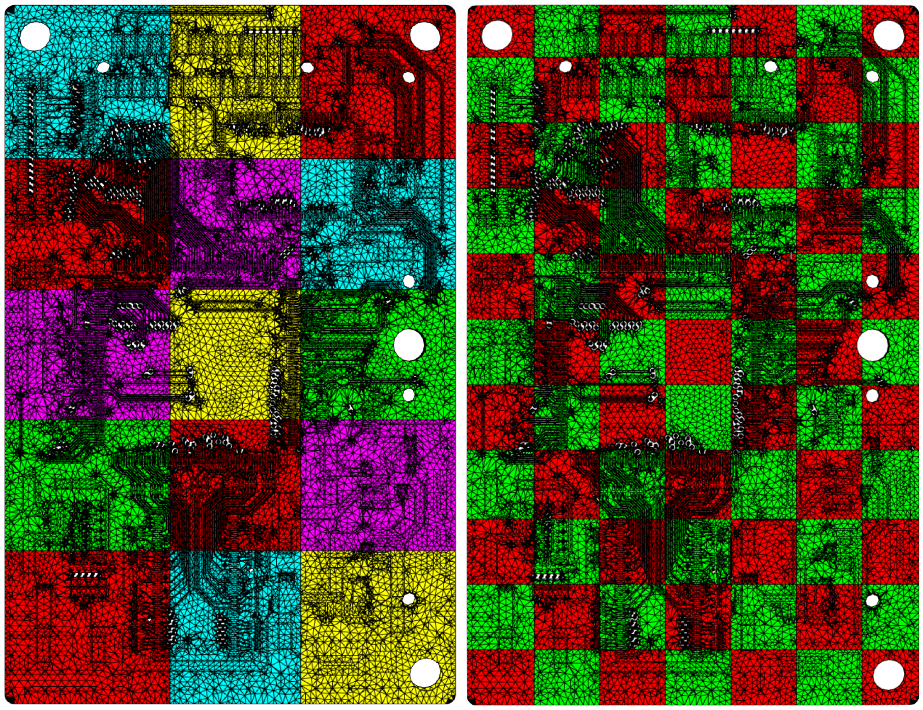


Figure 1.6. Domain partition of a printed circuit board model (Left: 15 domains; Right: 77 domains).

(2) Another factor that affects the computational efficiency arises from the DDM formulation. In general, the FETI method and all its variants require the computation of a domain discreet Dirichlet-to-Neumann (DtN) map or Steklov-Poincare operator [1] which is usually the most computationally expensive step in DD. The physical meaning of such DtN map shares a very similar spirit of the impedance matrix used in Method of Moment (MoM), where such matrix computation is usually accelerated by using adaptive cross approximation (ACA) methods [46–48]. Zhao et. al. extends the same idea to apply with the FETI-"like" FEM-DDM in [49]. Although it shows considerable efficiency improvement with the adaptive cross approximation (ACA) algorithm, it is known that the ACA method was originally designed for asymptotically smooth kernels [50], it would be questionable to apply in EM problems as the integral kernel in IEs is oscillatory. For the same matrix, Paraschos [1] used a selective inverse matrix computation approach to improve its computation efficiency at some extent, however the proposed approach does not take advantage of the low-rank properties of such matrices, and memory reduction was not achieved.

1.3.3 Randomized Algorithms for Matrix Approximation and Decomposition

Randomized algorithms are probably the most well known methods in numerical computation, and a very obvious and successful example is the Monte Carlo method [51] which represents a broad types of methods relying on repeated random sampling to obtain numerical results. Surprisingly even in the era of sufficient choice of sophiticated numerical algorithms, the Monte Carlo methods have become more and more important for modern computational applications [52].

In the last decade, randomized matrix algorithms [53] have attracted a significant amount of interest from different areas like machine learning [54], computational biology and bioinformatics [55] and etc. Such widespread interest mainly arose from

the need of effective and efficient methods to deal with masssive data set (or more specifically, large-scale matrices) that are being generated. The "randomized matrix algorithms" refer to a class of random sampling and projection methods for ubiquitous linear algebra problems such as matrix approximation and decomposition problems. Among all applications in linear algebra, we are particularly interested in fast matrix SVD and low-rank approximation that will be adopted in this dissertation work.

Johnson and Lindenstrauss [56] are arguably the first to incoporate randomized ideas to compress data which is similar to the works to compute the Latent Semantic Indexing proposed by Papadimitriou et. al. [57]. Martinsson et. al. [58] proposed a randomized algorithm for low-rank matrix approximation that firstly derived the error bounds and also introduced the idea of over sampling for further improvement. Following the same line, Halko et. al. [5] proposed a randomized SVD computation framework that is a better version of [58] that combines the random projection method [59] and power iteration to achieve a more efficient random sampling scheme.

1.4 Contributions of Proposed Work

In this work, we aim to propose a single CEM method that is suitable for robust and effective computation and parallel processing, thus capable of harnessing the extra power of multi/many core computers, high performance computing (HPC), cloud computing, etc. The review of previous work revealed that a hybrid of domain decomposition methods (DDM) [20] and finite element method (FEM) [60] appears to be ideal. As on one hand, FEM is a very general solver for solving problems with arbitrary geometries and material properties, and on the other hand, DDM is ideal for parallel computations. This combination will require minimal input or expertise from the users. Following this approach, as shown in Fig. 1.7, the state-of-art is the MG-FETI-LEAP proposed in [1] which shows significant advantages over its competitors. However due to the ad-hoc nature of selected global basis functions and intensive

computation required to solve domain numerical Green's function, MG-FETI-LEAP still suffers from the lack of enough numerical robustness and efficiency for large-scale and complicated structures, which will be discussed in more details in the upcoming review sections and results chapter. Therefore, to obtain a FEM-DD framework that is *numerically robust* and *efficient* enough while maintaining the optimal *parallel scalability* of DDM is still a challenge.

As it become clear from the literature review, the prevalent approaches at improving computational efficiency and possibly reliability of electromagnetic computations either combine different CEM methods in the form of hybrid methods, or combine domain decomposition ideas with advanced physical or discretization or even linear algebra concepts. The common denominator of all these approaches is that they rely on purely deterministic computations and require significant user intuition to suitably choose the various physical or computational approximations and parameters. In contrary, this work aims to reduce the reliance of CEM computations on expert-user choices by combining deterministic and randomized algorithms under the paradigm of domain decomposition. A high level overview of the proposed DDM framework is described in Fig. 1.8.

The breakthrough in achieving the combination of these seemingly disparate methods, i.e., deterministic and stochastic methods, came from studying the properties of the discrete Dirichlet-to-Neumann (DtN) map operation that is involved in the computation of each FETI-2λ interface equation. The discrete DtN map operator behaves as the numerical Green's function for each domain in the decomposition, that represents the electromagnetic interaction of the interface degrees-of-freedom (DoFs), i.e. EM wave transmitters and receivers distributed at the domain interface, in the presences of the geometry and materials included in the volume of the domain. Since these interactions represent sources and receivers radiating in an EM environment governed by Maxwell's equations with known boundary conditions, they could be grouped into

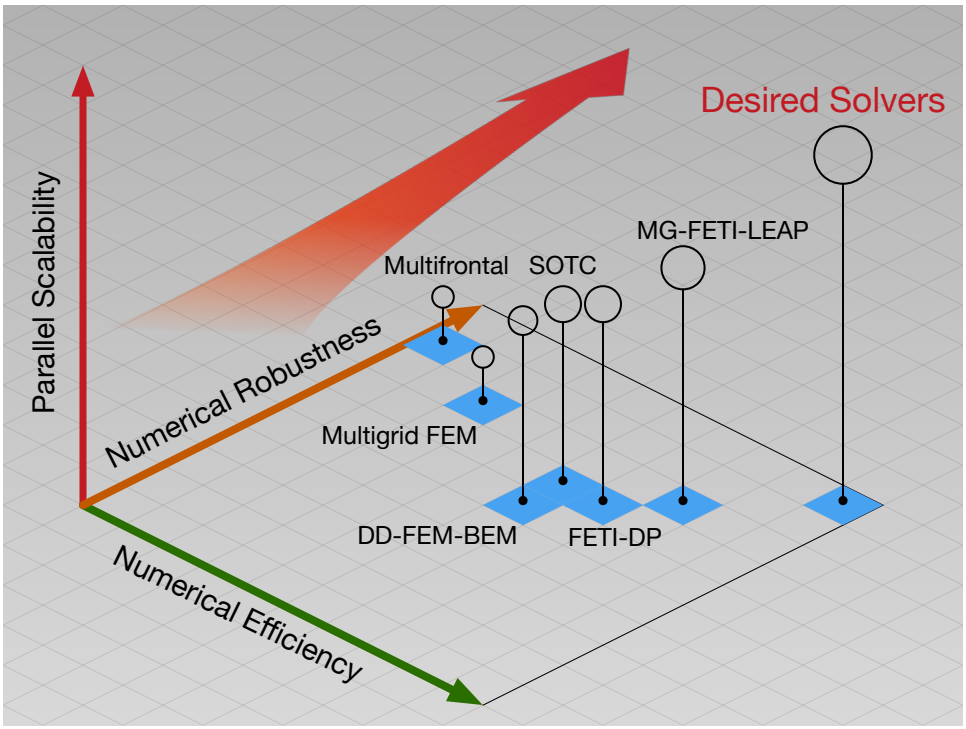


Figure 1.7. A visual comparison of finite elements based solvers for multiscale computational EM problems in terms of numerical robustness, numerical efficiency and parallel scalability.

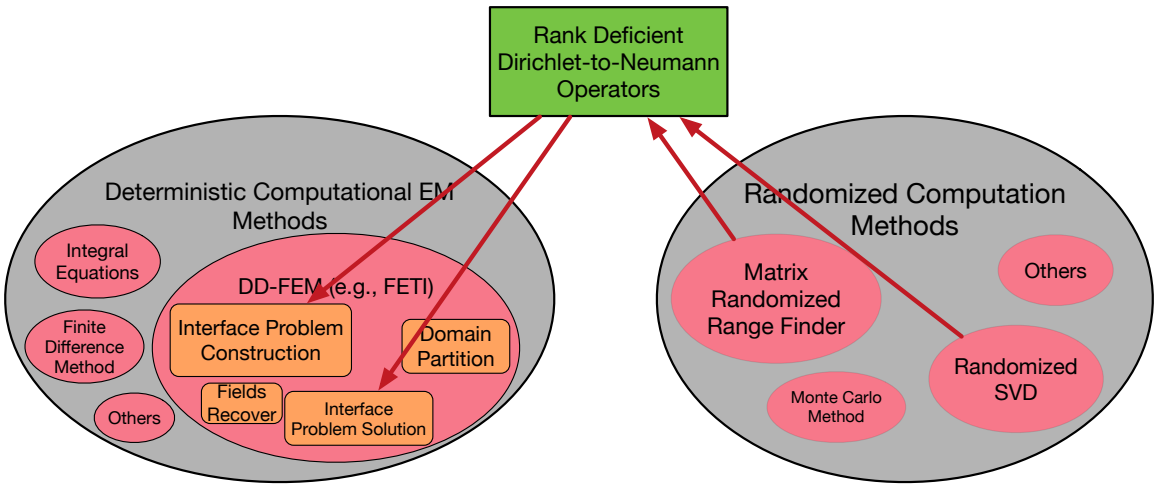


Figure 1.8. A overview of proposed DDM framework in conjunction with randomized algorithms.

three types based on the physics of these interactions: 1) singular, self interactions, when source and observer coincide; 2) strong coupling or near-singular interactions, when source and observer are located electrically close; 3) weak coupling, low-rank interactions, when source and observer are well separated. It is very hard to use randomization ideas for former two cases, as their computation is sensitive to small, even round-off errors. However it is possible to use randomization ideas to compute and represent low-rank interactions, as well as preconditioners in iterative solution schemes. More specifically this work will propose:

1. FETI-2λ DD method, that uses a randomized range finder and low-rank approximation to speed-up the computation of interface equation and reduce the methods memory footprint. The proposed work capitalizes on the numerical low-rank property of the domain DtN operators mapping matrix involved in the FETI-2λ formulation to save memory and time for its computation. More specifically, it combines a randomized singular value decomposition computation with matrix selective inverse solver (e.g., MuMPS) and a geometrical partitioning tree to efficiently compute the discrete DtN map operator in each domain.

Improvements in memory and run-time for up to 30% - 50% have been obtained for a variety of problem types, sizes and domain decompositions.

2. A effective, efficient and reliable global preconditioner that uses the Woodbury block matrix inversion formula [61] that internally relies on randomized singular value decomposition (R-SVD) method for its efficiency. The Woodbury preconditioned based FETI or W-FETI is algebraic and uses the dominant subspace of the off-diagonal (neighboring) interaction in the FETI interface equation to form a global problem that is then directly factorized and applied to the FETI-2λ system. W-FETI elegantly and automatically detects the nature of the underlying physical interactions by SVD-based data mining the discrete DtN operators of all domains, therefore is error-controllable and is optimal in size. The preconditioner is applicable to arbitrary decompositions with conforming and non-conforming meshes along the domain interface which greatly benefit users for meshing flexibility. The global preconditioner is later integrated with local accelerators - LEAP^p in a multiplicative fashion, leading to a near optimal preconditioning scheme that enhance the efficiency and reliability/resilience for FETI-2λ DDM. In fact, W-FETI is a generalization of FETI-LEAP since the the first level local preconditioner LEAP^1 comes naturally from the diagonal block part of W-FETI. Numerical experiments and real-life examples quantify siginificant memory and run-time savings over the state-of-the-art.

3. A W-FETI DD in conjunction with randomized domain discrete DtN computation without considerably sacraficing parallel scalability. The combination of the randomized discrete DtN computation and W-FETI global preconditioner is expected to significantly reduce the memory and run-time overhead and at the same time, offers a reliable and efficient global preconditioning scheme. In this work, careful interface indexing is implemented to make the discrete DtN com-

putation and many manageable-size SVD in W-FETI become a embarassingly parallel procedure, meaning no communication overhead is needed.

1.5 Dissertation Outline

The rest of the dissertation is organized as follows.

Chapter 2 briefly reviews some preliminary theories and background knowledge to support this research. It introduces basic EM theories with some function spaces and notations. A general EM model is given to demonstrate the FETI-2λ domain decomposition finite element formulation. This chapter also introduces important EM engineering quantities that will be used throughout the dissertation manuscript.

Chapter 3 develops the core linear algebra algorithm that will be used throughout this work - matrix randomized rank range finder and singular value decomposition (RSVD) algorithm. It then introduces a new approach to accurately and efficiently compute the domain discrete DtN map operators. The RSVD algorithm will be exploited to leverage expediating and compressing the far-separated interactions of the DtN map. Two different numerical examples are given followed by the theory part to validate the accuracy and computational resource saving with the proposed algorithm.

Chapter 4 introduces a global preconditioning technique that promises to restore the numerical scalability of FETI-2λ. Starting from the established FETI-2λ formulation, the new global preconditioning is derived using clever matrix re-partitioning and well-known Woodbury matrix identity. It then introduces two important techniques that enable to efficiently conduct many manageable-size SVD rather than performing a computationally prohibitive SVD on the original FETI matrix. The integration of local LEAP preconditioner and W-FETI global preconditioner is also presented. A set of numerical studies will be given after the theory to show the numerical scalability of

the proposed preconditioner w.r.t. domain number, domain size, mesh discretization and with structured/unstructured meshing strategy.

Chapter 5 tests the proposed FETI-2λ DDM framework on five different very challenging, realistic and multiscale EM problems. Namely, a X-band waveguide filter, two multi-layer printed circuit board, a finite Vivaldi array enclosed with a radome and a generic drone aircraft. Results comparison in terms of accuracy, efficiency and reliability between the proposed DDM framework with the state-of-the-art are presented.

Chapter 6 ends the dissertation with a summary section and conclusions.

CHAPTER 2

PRELIMINARIES

This chapter introduces notations, basic time-harmonic electromagnetics equations and the underlying finite element tearing and interconnecting with two Lagrange multipliers (FETI-2λ) domain decomposition method that this dissertation aims to improve.

2.1 Notations

Boldface capital letters (e.g. $\mathbf{S}$) are used to represent matrices in $\mathbb{C}^{N\times N}$, whereas boldface lowercase letters (e.g. $\mathbf{u}$) represent Euclidean vectors or vector fields in $\mathbb{C}^3$ or vectors in $\mathbb{C}^N$. An overhead hat (e.g. $\hat{\mathbf{u}}$) represents a unit vector. Caligraphic letters are used to denote geometric, decomposition or mesh related variables, e.g. $\mathcal{M}$ - mesh, $\mathcal{K}$ - tetrahedron, $\mathcal{E}$ - edge, $\mathcal{F}$ - face, $\mathcal{V}$ - vertix.

Script letters such as $\boldsymbol{\mathscr{E}}$ and $\boldsymbol{\mathscr{H}}$ represent real and time varying electric and magnetic vector fields that correspond to the time-harmonic electric $\mathbf{E}$ and magnetic $\mathbf{H}$ vector fields. Wave number in free-space is denoted by $k_o = \omega\sqrt{\mu_0\epsilon_o}$, where $\omega = 2\pi f$ is the radial frequency with f represents the operational frequency, μ_o and ϵ_o represent the material permeability and permitivity in free-space, respectively. In any dielectric object, the material permeability and permitivity are defined as $\mu = \mu_r\mu_o$ and $\epsilon = \epsilon_r\epsilon_o$, respectively.

The imaginary unit symbol used in this disseration is j and the $e^{j\omega t}$ time-harmonic convention is followed.

2.2 Electromagnetic Theory Review

This section introduces some fundamental electromagnetic quanties and the governing Maxwell's equations that lead to the proposed boundary value problem in the next section. Finally, common EM simulation engineering metrics and quantities such as s-parameters and far-field pattern are derived at the end of the section.

2.2.1 Maxwell's Equations

In EM, the time-varying quantities we are interested in general are:

$$\begin{gathered}
\boldsymbol{\mathcal{E}}(\mathbf{r},t) : \text{Electric field intensity,} \\
\boldsymbol{\mathcal{D}}(\mathbf{r},t) : \text{Electric flux density,} \\
\boldsymbol{\mathcal{H}}(\mathbf{r},t) : \text{Magnetic field intensity,} \\
\boldsymbol{\mathcal{B}}(\mathbf{r},t) : \text{Magnetic flux density,} \\
\mathbf{E}(\mathbf{r},\omega) : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{C}^3 : \text{Time-harmonic electric field intensity.}
\end{gathered}$$

where $\mathbf{r}$ represents a position vector in space, t denotes time. These vector fields are governed by Maxwell's equations:

$$\begin{aligned}
\oint_C \boldsymbol{\mathcal{E}} \cdot d\mathbf{l} &= -\frac{d}{dt} \iint_S \boldsymbol{\mathcal{B}} \cdot d\mathbf{s}, \\
\oint_C \boldsymbol{\mathcal{H}} \cdot d\mathbf{l} &= \frac{d}{dt} \iint_S \boldsymbol{\mathcal{D}} \cdot d\mathbf{s} + \iint_S \boldsymbol{\mathcal{J}} \cdot d\mathbf{s}, \\
\oiint_S \mathbf{B} \cdot d\mathbf{s} &= 0, \\
\oiint_S \boldsymbol{\mathcal{D}} \cdot d\mathbf{s} &= \iiint_V q_v \, dv.
\end{aligned} \tag{2.1}$$

where $\boldsymbol{\mathcal{J}}$ is the impressed density of free currents, and q_v denotes the volume density of free charges, and the space-coordinate and time arguments have been omitted for

simplicity. The circle on a line integral denotes a closed contour while the circle on a surface integral denotes a closed surface.

Not all unknown quantities in (2.1) are independent, and not all equations either. The relationship between field intensity and flux density can be easily established by the constitutive equations,

$$\begin{aligned} \boldsymbol{\mathcal{D}} &= \epsilon * \boldsymbol{\mathcal{E}}, \\ \boldsymbol{\mathcal{B}} &= \mu * \boldsymbol{\mathcal{H}}, \end{aligned} \tag{2.2}$$

where ϵ and μ can in general be the permitivity and permeability tensors, respectively for the wave propagation medium. $*$ denotes a convolution operation. In this dissertation, the special case of the isotropic medium will be considered, thus these tensors become simple complex functions of position.

The Maxwell's equations in (2.1) can also be expressed in differential form by applying Stokes theorem on the first two equations in (2.1) and divergence theorem on the other two. This leads to

$$\begin{aligned} \nabla \times \boldsymbol{\mathcal{E}} &= -j\omega\boldsymbol{\mathcal{B}}, \\ \nabla \times \boldsymbol{\mathcal{H}} &= j\omega\boldsymbol{\mathcal{D}} + \boldsymbol{\mathcal{J}}, \\ \nabla \cdot \boldsymbol{\mathcal{B}} &= 0, \\ \nabla \cdot \boldsymbol{\mathcal{D}} &= q_v. \end{aligned} \tag{2.3}$$

Up till now, the field intensity and flux density quantities are all represented in the form of *instantaneous quantities*. For instance, the electric field intensity is expressed as

$$\boldsymbol{\mathcal{E}} = \sqrt{2}\mathrm{Re}(\mathbf{E}\, e^{jwt}), \tag{2.4}$$

where $\mathbf{E}(\mathbf{r}, \omega) : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{C}^3$ represents the time-harmonic electric field intensity. A scalor $\sqrt{2}$ is used to denote that $\mathbf{E}$ is a root-mean-square (rms) value. Following such simplification, (2.3) can be rewritten as

$$\begin{aligned} \nabla \times \mathbf{E} &= -j\omega\mathbf{B}, \\ \nabla \times \mathbf{H} &= j\omega\mathbf{D} + \mathbf{J}, \\ \nabla \cdot \mathbf{B} &= 0, \\ \nabla \cdot \mathbf{D} &= q_v. \end{aligned} \tag{2.5}$$

with well-known boundary conditions,

$$\begin{aligned} \mathbf{E}_t &= 0 \quad \text{on } \Gamma_{PEC} \\ \mathbf{H}_t &= 0 \quad \text{on } \Gamma_{PMC} \\ \lim_{\mathbf{r}\to\infty} |\mathbf{r}| &\left(\hat{\mathbf{r}} \times \mathbf{H} + \frac{1}{\eta}\mathbf{E}\right) = 0 \end{aligned} \tag{2.6}$$

where $\mathbf{E}_t$ and $\mathbf{H}_t$ denote the tangential component of the complex vector field quantity on the surface of a perfect electric/magnetic conductor, respectively. The last equation is the Silver-Müller radiation condition [62] imposed at infinity in unbounded problems.

2.2.2 EM Engineering Quantities

Very often, the final result of an electromagnetic simulation is an engineering quantity such as the device/system scattering matrix, and/or the far field quantities (e.g., radiation pattern of an antenna or scattering pattern from an object). In the next two sections, we define these commonly used EM engineering quantities and show how they are related to the electric and magnetic fields solved by the proposed methods.

2.2.2.1 s-parameters

The scattering matrix relates the incident and refective voltage waves at the ports of a device [63]. For a multi-port microwave devices or antennas, these quantities can usually be directly measured by a modern vector network analyzer device.

Consider a P-port network, then the entry at (i, j) of the scattering matrix S is defined as below,

$$\mathbf{S}_{i,j} = \frac{V_i^-}{V_j^+} \quad \text{where } V_k^+ = 0, k \neq j, \tag{2.7}$$

where V_j^+ denotes the amplitude of the voltage wave incident on port j, while V_i^- denotes the amplitude of the voltage wave reflected from port j at port i. It is important to note that to measure $\mathbf{S}_{i,j}$, all the other ports other than i and j need to be terminated to a matched load, i.e. $V_k^+ = 0$. The voltage at port k can be calculated using

$$V_k^- = \int_{\Gamma_k} \hat{\mathbf{n}} \times \mathbf{E} \times \hat{\mathbf{n}} \times \mathbf{J}_k^{\text{modal}} \, dr^2 \tag{2.8}$$

where $\mathbf{J}_k^{\text{modal}} = \hat{\mathbf{n}} \times \mathbf{H}_k^{\text{modal}}$, $\hat{\mathbf{n}}$ is the normal unit vector to the port, and $\mathbf{H}_k^{\text{modal}}$ is the propagating waveguide modal field of the k^{th} port, Γ_k represents the boundary surface at that port.

2.2.2.2 EM Far Fields

Another important EM engineering quantity of interest such as antenna gain, radiated power or radar cross section (RCS) are derived from far-fields (fields evaluated at observations at infinity). FETI-2λ solves for the near-field quantity$\mathbf{E}$ thus an extra near-to-far field transformation step must be invoked using the far-field representation of the Stratton-Chu formula [64],

$$\mathbf{E}_\infty(\hat{\mathbf{r}}) = -\frac{jk}{4\pi} \oint_{\partial\Omega} \left(\eta \mathbf{J}(\mathbf{r}') + \mathbf{M}(\mathbf{r}') \times \hat{\mathbf{r}} \right) e^{jk\hat{\mathbf{r}}\cdot\mathbf{r}'} \, d\mathbf{r}'^2 \tag{2.9}$$

where

$$\begin{aligned} \mathbf{J}(\mathbf{r}') &= \hat{\mathbf{n}}' \times \mathbf{H}(\mathbf{r}'), \\ \mathbf{M}(\mathbf{r}') &= -\hat{\mathbf{n}}' \times \mathbf{E}(\mathbf{r}') \end{aligned} \tag{2.10}$$

are the electric and magnetic currents on the computation domain bounding surface $\partial\Omega$, respectively. $\hat{\mathbf{r}}$ is the observation position unit vector, and $\mathbf{r}'$ is the position vector of the source fields, at the surface $\partial\Omega$. The radiation intensity is now defined as

$$U(\hat{\mathbf{r}}) = \frac{r^2}{\eta}|\mathbf{E}_\infty(\mathbf{r},\theta,\phi)|^2. \tag{2.11}$$

Thus, the antenna directivity can be shown as

$$D(\hat{\mathbf{r}}) = 4\pi\frac{U(\hat{\mathbf{r}})}{P_{rad}}, \tag{2.12}$$

where the radiated power is

$$P_{rad} = \int_{\theta=0}^{\pi}\int_{\phi=0}^{2\pi} U(\hat{\mathbf{r}},\theta,\phi)\sin\theta\, d\theta d\phi. \tag{2.13}$$

For scattering problems, the RCS is given as

$$A_e(\hat{\mathbf{r}},\mathbf{k}_{inc}) = \lim_{r\to\infty} 4\pi\frac{|\mathbf{E}_\infty^{scatt}(\hat{\mathbf{r}})|^2}{|\mathbf{E}^{inc}(\mathbf{k}_{inc})|^2}. \tag{2.14}$$

where $\mathbf{r} = \hat{\mathbf{r}}r$ and $\mathbf{E}^{inc}(\mathbf{k}_{inc})$ represents the incident electric field.

2.3 Boundary Value Problem Statement

Before defining the problem statement, it would be useful to introduce some space which will be necessary for the development of FETI-2λ DDM formulation. The space of tangentially continuous vector fields, such as $\mathbf{E}$ in Ω is

$$\mathbf{H}(\mathbf{curl};\Omega) = \{\mathbf{u}\in\mathbf{H}(\mathbf{curl};\Omega)\,|\,\nabla\times\mathbf{u}\in(\mathcal{L}_2(\Omega))^3, \mathbf{u}\in(\mathcal{L}_2(\Omega))^3\}, \tag{2.15}$$

where $\mathcal{L}_2(\Omega)$ represents a set of square integrable functions over the computational domain Ω. Further, we define vector field space $\mathbf{H}_\circ(\mathbf{curl};\Omega)$ as

$$\mathbf{H}_\circ(\mathbf{curl};\Omega) = \{\mathbf{u} \in \mathbf{H}_\circ(\mathbf{curl};\Omega) \,|\, \mathbf{u} \in \mathbf{H}(\mathbf{curl};\Omega), \hat{\mathbf{n}} \times \mathbf{u} = 0 \text{ on } \Gamma_{\text{pec}}\}, \tag{2.16}$$

in other words, if the elements of $\mathbf{H}(\mathbf{curl};\Omega)$ represent the electric fields, then the subspace of $\mathbf{H}_\circ(\mathbf{curl};\Omega)$ represents electric fields that satisfy the EM boundary conditions on PECs.

A generic EM system with various materials, excitations and boundary conditions is shown in Fig. 2.1. The EM analysis of such is formally casted into the following boundary value problem (BVP) statement:

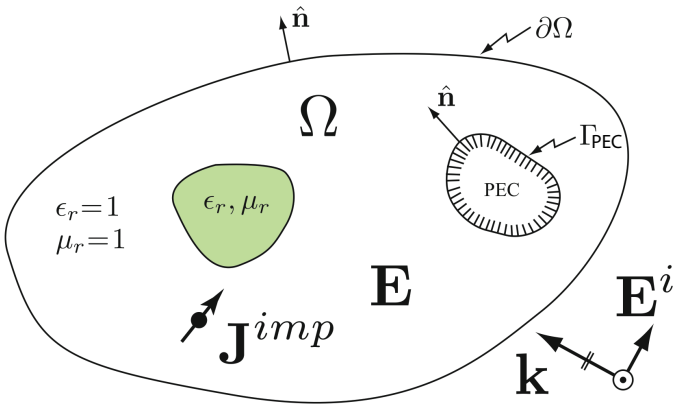


Figure 2.1. A generic EM system used for the development of the boundary value problem.

$$\begin{gathered}
\textit{Seek } \mathbf{E} \in \mathbf{H}_\circ(\mathbf{curl};\Omega) \textit{ such that} \\
\nabla \times \frac{1}{\mu_r} \nabla \times \mathbf{E} - \epsilon_r k_o^2 \mathbf{E} = -j\omega\mu_0 \mathbf{J}_{imp}, \quad \text{in } \Omega, \\
\hat{\mathbf{n}} \times \frac{1}{\mu_r} \nabla \times \left(\mathbf{E} - \mathbf{E}^{inc}\right) + jk_o\eta \hat{\mathbf{n}} \times \left(\mathbf{E} - \mathbf{E}^i\right) \times \hat{\mathbf{n}} = 0, \quad \text{on } \partial\Omega,
\end{gathered} \tag{2.17}$$

where most variables are defined in section 2.1, η is the wave impedance given by $\eta = \sqrt{\epsilon_r/\mu_r}$, $\mathbf{J}^{imp}$ is a given port excitation current used in radiation or driven guided-wave problems, whereas $\mathbf{E}^{inc} = \mathbf{E}\, e^{-j\mathbf{k}_{inc}\cdot\mathbf{r}}$ is the given incident transverse electric and

magnetic TEM plane wave field propagating in the $\mathbf{k_i} = k_0\hat{\mathbf{k}}_i$. It is noted that the last equation in Eq. (2.17) refers to the first order absorbing boundary condition (ABC) [23] applied at the truncation surface $\partial\Omega$, and is used to emulate the behavior of unbounded free-space.

2.4 FETI-2λ DDM

The proposed work is based on a FETI-2λ DDM presented in [1] that would be summarized here for sake of completeness. The FETI-2λ is a non-overlapping DDM of the finite element tearing and interconnecting family [25]. It uses two sets of Lagrange Multipliers (LMs) at each domain interface, and impedance type transmission conditions [42].

Before the theory development, let us introduce a new notation, a subscript $*$ denoting the decomposed fields, namely fields that are tangentially continous throughout the domain interior but can be discontinuous across them. The decomposed electric fields $\mathbf{E}_* \in \mathbf{V}_* = \prod_{i=1}^{N} \mathbf{H}_o(\mathbf{curl}; \Omega_i)$ where i represents the domain index. In FETI-2λ DDM, the LMs are defined as

$$\boldsymbol{\lambda} = \mathbf{j} + \alpha\mathbf{e} \ \in \ \boldsymbol{\Lambda}, \tag{2.18}$$

where $\mathbf{j} = \hat{\mathbf{n}} \times \mathbf{H}$ is the surface electric currents, α is a complex scalar that is set to jk, $\mathbf{e} = \hat{\mathbf{n}} \times \mathbf{E} \times \hat{\mathbf{n}}$ is the surface electric field, and $\boldsymbol{\lambda}$ is a space of tangentially continuous functions with interfaces alone but discontinuous between different interfaces,

$$\boldsymbol{\Lambda}_* = \prod_{i=1,j=1}^{K} \mathbf{H}^{-\frac{1}{2}}(\mathbf{curl}_\Gamma; \mathcal{I}_{ij}), \tag{2.19}$$

where $\mathcal{I}$ denotes the domain interface between domain i and j. It is noted that this definition of interfaces allows for interfaces that are duplicate, i.e $\mathcal{I}_{12}$ and $\mathcal{I}_{21}$, but that is done purposely to easier account for non-conforming meshes across domains.

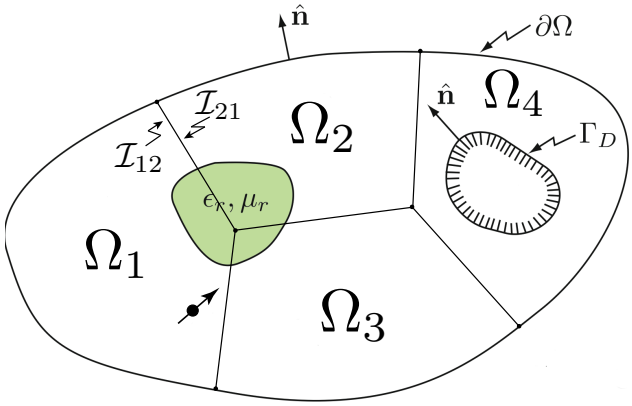


Figure 2.2. A geometry used for the development of the decomposed boundary value problem.

The decomposed boundary value problem (DBVP) based on the decomposed computational problem as shown in Fig. 2.2 in terms of $\boldsymbol{\lambda}$ reads as,

$$\begin{aligned} &\textit{Seek } (\mathbf{E}_*, \boldsymbol{\lambda}) \in \{\mathbf{V}_*, \boldsymbol{\Lambda}\} \textit{ such that} \\ \nabla \times \frac{1}{\mu_r} \nabla \times \mathbf{E}_* - \epsilon_r k_o^2 \mathbf{E}_* &= -j\omega\mu_0 \mathbf{J}_{imp}, \quad \text{in } \Omega, \\ \mathbf{R}_{ij}\boldsymbol{\lambda} + \mathbf{R}_{ji}\boldsymbol{\lambda} - 2\alpha \mathbf{R}_{ji}\mathbf{e} &= 0, \quad \text{on } \mathcal{S}_{ij}, [i,j] = [1, \cdots, N], \\ \mathbf{n} \times \frac{1}{\mu_r} \nabla \times \left(\mathbf{E} - \mathbf{E}^i\right) + jk_o\sqrt{\eta}\mathbf{n} \times \left(\mathbf{E} - \mathbf{E}^i\right) \times \hat{\mathbf{n}} &= 0, \quad \text{on } \partial\Omega, \end{aligned} \tag{2.20}$$

The second equation represents the impedance type TC [15] with respect to the LM $\boldsymbol{\lambda}$, $\mathcal{S} = \bigcup \mathcal{I}_{ij}$ denotes the domain-face skeleton arising from the decomposition of the computational domain Ω, N is the total domain count, and $\mathbf{R}_{ij}$ is a restriction operator: $\mathbb{C}^3(\mathcal{S}) \rightarrow \mathbb{C}^3(\mathcal{I}_{ij})$ where $\mathbf{u}_{ij}^d = \mathbf{R}_{ij}\mathbf{u}^d$. Here, $\mathbf{u}^d$ represents the vector field in any domain d, and $\mathbf{u}_{ij}$ represents its component on a particular interface $\mathcal{I}_{ij}$. For the decomposition given in Fig. 2.2, the restriction operator is given as follows,

$$\begin{bmatrix} \mathbf{R}_{12} \\ \mathbf{R}_{13} \\ \mathbf{R}_{21} \\ \mathbf{R}_{23} \\ \mathbf{R}_{24} \\ \mathbf{R}_{31} \\ \mathbf{R}_{32} \\ \mathbf{R}_{34} \\ \mathbf{R}_{42} \\ \mathbf{R}_{43} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{2.21}$$

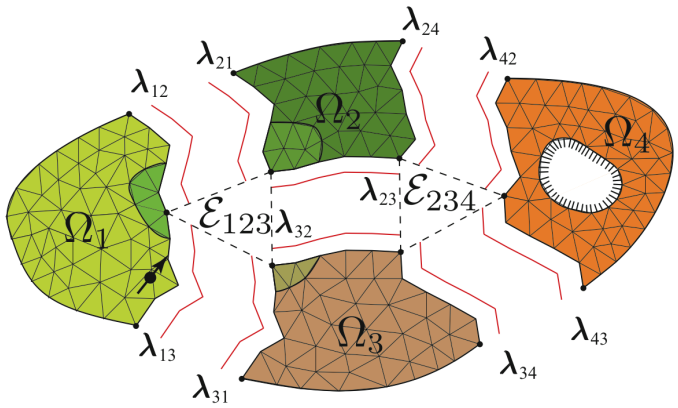


Figure 2.3. A computational model with 4 domains in an unstructured, paritioning topology.

Without going into the details of the variational re-casting, Galerkin discretization and finite element approxiation, the final matrix representation of FETI-2λ reads as,

$$\begin{bmatrix} \mathbf{A}_1 & \dots & \mathbf{0} & \mathbf{D}_1 \\ \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \dots & \mathbf{A}_N & \mathbf{D}_N \\ \mathbf{B}_1 & \dots & \mathbf{B}_N & \mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_N \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_N \\ \mathbf{0} \end{bmatrix} \tag{2.22}$$

where the unknown vectors $\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_N$ represent electric field DoFs on each domain, each domain is employed with a right hand side $\mathbf{f}_i$ whose formula can be found in [1], the matrix form are given as

$$\begin{aligned} \mathbf{A}_i(m,n) &= a_*(\mathbf{w}_{im}, \mathbf{w}_{in}), \\ \mathbf{D}_i(m,n) &= d(\mathbf{w}_{im}, \tilde{\mathbf{w}}_n), \\ \mathbf{B}_i(m,n) &= b(\tilde{\mathbf{w}}_m, \mathbf{w}_{in}), \\ \mathbf{T}(m,n) &= t(\tilde{\mathbf{w}}_m, \tilde{\mathbf{w}}_n), \end{aligned} \tag{2.23}$$

$\mathbf{w}$ and $\tilde{\mathbf{w}}$ are the basis functions defined in the primal and LM space to approximate the discrete fields $\mathbf{E}_*^h$ and $\boldsymbol{\lambda}^h$ with unknown coefficient set $\{e\}$ and $\{\lambda\}$. Therefore, the unknown vector in (2.22) is expressed as $\mathbf{e}_i = [e_{i1}, e_{i2}, \cdots, e_{in_i}]^{\mathrm{T}}$ and $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \cdots, \lambda_{n_\lambda}]^{\mathrm{T}}$ where n_i denotes the number of primal unknowns in ith domain, and n_λ is the total number of dual unknowns in the decomposition. The various sesquilinear and bilinear operators in (2.23) are,

$$\begin{aligned} a_*(\mathbf{v}_1, \mathbf{v}_2) &= \sum_{i=1}^{N} a_i(\mathbf{v}_1, \mathbf{v}_2) - \alpha t_*(\mathbf{v}_1, \mathbf{v}_2) && \mathbf{V}_* \times \mathbf{V}_* \to \mathbb{C}, \\ d(\mathbf{v}, \boldsymbol{\lambda}) &= \int_{\mathcal{S}} \hat{\mathbf{n}} \times \mathbf{v} \times \hat{\mathbf{n}} \cdot \boldsymbol{\lambda}\, dr^2 && \mathbf{V}_* \times \boldsymbol{\Lambda}_* \to \mathbb{C}, \\ b(\boldsymbol{\lambda}, \mathbf{v}) &= -2\alpha \int_{\mathcal{S}} \boldsymbol{\lambda} \cdot \hat{\mathbf{n}} \times \mathbf{v} \times \hat{\mathbf{n}}\, dr^2 && \boldsymbol{\Lambda}_* \times \mathbf{V}_* \to \mathbb{C}, \\ t(\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2) &= \int_{\mathcal{S}} \boldsymbol{\lambda}_1 \cdot \boldsymbol{\lambda}_2\, dr^2 && \boldsymbol{\Lambda}_* \times \boldsymbol{\Lambda}_* \to \mathbb{C}, \end{aligned}$$

where

$$a_i(\mathbf{v}, \mathbf{u}) = \int_{\Omega_i} \left(\nabla \times \mathbf{v} \cdot \frac{1}{\mu_r} \bigtriangledown \times \mathbf{u} - k^2 \mathbf{v} \cdot \epsilon_r \mathbf{u} \right) dr^3 + jk \int_{\partial\Omega_i} \hat{\mathbf{n}} \times \mathbf{v} \cdot \hat{\mathbf{n}} \times \mathbf{u}\, dr^2,$$

$$t_*(\mathbf{v}_1, \mathbf{v}_2) = \int_{\mathcal{S}} \hat{\mathbf{n}} \times \mathbf{v}_1 \cdot \hat{\mathbf{n}} \times \mathbf{v}_2\, dr^2.$$

The original FETI-2λ DDM system in (2.22) can be reduced to a non-symmetric linear system acting on the dual (LM) unknowns only (that would be termed, discrete

interface equations), through the formation of the Schur complement (elimination of the primal unknowns). The discrete interface equations can be written as,

$$\mathbf{F}\boldsymbol{\lambda} = \mathbf{g}, \tag{2.24}$$

where

$$\begin{aligned} \mathbf{F} &= \mathbf{T} - \sum_{i=1}^{N} \mathbf{B}_i \mathbf{A}_i^{-1} \mathbf{D}_i, \\ \mathbf{g} &= -\sum_{i=1}^{N} \mathbf{B}_i \mathbf{A}_i^{-1} \mathbf{f}_i. \end{aligned} \tag{2.25}$$

The non-symmetric linear system in (2.24) is solved iteratively with the use of Krylov solvers, local and global preconditiong techniques described in [1]. The Krylov solver used in this work is the Induced Dimension Reduction (IDR(s)) [65, 66] solver with $s = 1$, except when stated otherwise. It is important to see that the elimination of primal unknowns essentially converts the DDM problem into an effective boundary element method (BEM) problem where $\mathbf{A}_i^{-1}$ takes the place of the numerical Green's function of the ith domain. It is also worth noting that the FETI-2λ formulation in (2.25) allows embarrassingly parallel, i.e. paralleization without any need for communication, computation of $\mathbf{A}^{-1}$ which makes the FETI-2λ DDM very efficient.

CHAPTER 3

RANDOMIZED COMPUTATION OF DIRICHLET-TO-NEUMANN MAP OPERATORS

This chapter proposes an randomized computational framework for fast and memory efficient assembly of the discrete Dirichlet-to-Neumann (DtN) map interactions that arise in FETI-2λ DDM by leveraging its numerical low-rank.

The proposed method in this chapter follows the same line of thought as described in [46] where a rank-revealing adaptive cross approximation (ACA) method was used to compress the discrete DtN map of a FETI-like interface equation. However this work adopts randomized algorithm which shows faster and significiantly more robust performance than ACA and allow for savings in both memory and run-time. The randomized algorithms presented here share the similar spirit with the seminal work of Halko, Martinsson and Tropp in [5]. The key innovation and computational improvement in this chapter arises from the combination of randomized rank-revealing computations with fast selective inverse direct solution strategies.

3.1 DtN Map Operators in FETI-2λ DDM

To facilitate the theory development, we consider the same decomposition example with mesh discretization shown in Fig.2.3. To fully appreciate the numerical properties of FETI-2λ and elucidate the insights that led to the development of this chapter, the matrix form of (2.24) can be explicited written as,

$$\left[\begin{array}{cc|cc|cc|cc|cc}
\mathbf{T}_{12} & \mathbf{Q}_{12,21} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{12,23} & \mathbf{0} & \mathbf{K}_{12,24} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{Q}_{21,12} & \mathbf{T}_{21} & \mathbf{K}_{21,13} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\hline
\mathbf{0} & \mathbf{0} & \mathbf{T}_{13} & \mathbf{Q}_{13,31} & \mathbf{0} & \mathbf{K}_{13,32} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{13,34} & \mathbf{0} \\
\mathbf{K}_{31,12} & \mathbf{0} & \mathbf{Q}_{31,13} & \mathbf{T}_{31} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{23,31} & \mathbf{T}_{23} & \mathbf{Q}_{23,32} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{23,34} & \mathbf{0} \\
\mathbf{0} & \mathbf{K}_{32,21} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{32,23} & \mathbf{T}_{32} & \mathbf{K}_{32,24} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{T}_{24} & \mathbf{Q}_{24,42} & \mathbf{0} & \mathbf{K}_{24,43} \\
\mathbf{0} & \mathbf{K}_{42,21} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{42,23} & \mathbf{0} & \mathbf{K}_{42,24} & \mathbf{T}_{42} & \mathbf{0} & \mathbf{0} \\
\hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{34,42} & \mathbf{T}_{34} & \mathbf{Q}_{34,43} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{43,31} & \mathbf{0} & \mathbf{K}_{43,32} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{43,34} & \mathbf{T}_{43}
\end{array}\right]
\left[\begin{array}{c}
\boldsymbol{\lambda}_{12} \\ \boldsymbol{\lambda}_{21} \\ \hline \boldsymbol{\lambda}_{13} \\ \boldsymbol{\lambda}_{31} \\ \hline \boldsymbol{\lambda}_{23} \\ \boldsymbol{\lambda}_{32} \\ \hline \boldsymbol{\lambda}_{24} \\ \boldsymbol{\lambda}_{42} \\ \hline \boldsymbol{\lambda}_{34} \\ \boldsymbol{\lambda}_{43}
\end{array}\right]
=
\left[\begin{array}{c}
\mathbf{g}_{12} \\ \mathbf{g}_{21} \\ \hline \mathbf{g}_{13} \\ \mathbf{g}_{31} \\ \hline \mathbf{g}_{23} \\ \mathbf{g}_{32} \\ \hline \mathbf{g}_{24} \\ \mathbf{g}_{42} \\ \hline \mathbf{g}_{34} \\ \mathbf{g}_{43}
\end{array}\right], \quad (3.1)$$

where all the off-diagonal sub-matrices that comprise the global matrix $\mathbf{F}$ involve

$$\mathbf{K}^{(d)}_{ij,mn} = -\mathbf{R}_{ij}\mathbf{B}^{(d)}\mathbf{A}^{-1}_{(d)}\mathbf{D}^{(d)}\mathbf{R}^{T}_{mn}, \quad (3.2)$$

where d denotes any given domain, and $\mathbf{B}, \mathbf{D}, \mathbf{R}$ were defined in (2.23). It is seen that evaluating (3.2) becomes the most computationally intensive step as it involves the numerical factorization of the domain FEM matrix $\mathbf{A}_{(d)}$ and it is encountered at every domain-to-domain interaction throughout the decomposition. Fortunately such matrix is never explicitly formed since in an iterative domain decomposition such as the FETI-2λ, only the matrix vector multiplication operation is sufficient. In [1], three different approaches were proposed for such computation and numerical studies suggested that the FETI-Z approach seems the most efficient one. In FETI-Z, (3.2) is rewritten as

$$\mathbf{K}^{(d)}_{ij,mn} = -(\mathbf{R}_{ij}\mathbf{B}^{(d)}\mathbf{R}^{eT}_{ji})(\mathbf{R}^{e}_{ji}\mathbf{A}^{-1}_{(d)}\mathbf{R}^{eT}_{mn})(\mathbf{R}^{e}_{mn}\mathbf{D}^{(d)}\mathbf{R}^{T}_{mn}), \quad (3.3)$$

where $\mathbf{R}^{e}_{ij} \in \mathbb{N}^{n^{e}_{ij} \times n_i}$ is a sparse restriction matrix that maps the vector of primal DoFs of the domain d to the primal DoFs at the interface $\mathcal{I}_{ij}$, i.e.,

$$\mathbf{e}^{(d)}_{ij} = \mathbf{R}^{e}_{ij}\mathbf{e}^{(d)} \quad (3.4)$$

where $\mathbf{e}_{ij}^{(d)}$ denotes the primal DoFs on interface $\mathcal{I}_{ij}$ of a given domain d and $\mathbf{e}^{(d)}$ denotes the volumetric primal DoFs. In (3.3), the first parenthesis term represents the sparse coupling matrix linking the LMs ($\boldsymbol{\lambda}$) on i^{th} interface with their duplicate set on its neighboring interface, $neigh(i)$. The last parenthesis term represents the sparse coupling matrix linking the LMs in j^{th} interface of domain d with the primal unknowns ($\mathbf{e}$) on the same interface. The middle term is an important matrix, usually termed as Z-matrix due to its equivalency to the impedance matrix encountered in PEC object BEM and represents the discrete representation of the domain DtN map. It can be defined for any two interfaces $\mathcal{I}_{ij}$ and $\mathcal{I}_{mn}$ pairs of domain d, therefore we can instead define a single z-matrix for domain d as,

$$\mathbf{Z}^{(d)} = \mathbf{R}^{e}_{(d)}\mathbf{A}^{-1}_{(d)}\mathbf{R}^{eT}_{(d)} \tag{3.5}$$

where $\mathbf{R}^{e}_{(d)} = \cup_{(ij)}\mathbf{R}^{e}_{(ij)}$. The Z-matrix representation of the discrete Dirichlet-to-Neumann (DtN) map of domain could be considered as the numerical Green's function of domain d for electric current sources and receivers located on the interfaces of the domain. The computation of (3.5) is a very intensive task as it involves a sparse matrix factorization that is an almost $\sim \mathcal{O}(N^2)$ operation where N is the dimension of the sparse matrix $\mathbf{A}_{(d)}$. Indeed, the computation of DtN operators is the most time-consuming step in the serial implementation of FETI-2λ DDM computation statistics suggests that is 75% $\sim$ 85% of the total run-time, but these time significantly reduce in parallel runs since this tasks are suitable for parallel computation.

In [1], an efficient approach was proposed in (3.5) forming via computing selective entries of $\mathbf{A}_{(d)}$ inverse using the direct solver MUMPS [67]. In this work, we propose a new approach that takes a totally different perspective and uses randomized algorithms to further improve the memory and run-time complexity of this computationally intensive step.

3.2 The Numerical Rank of the FETI-2λ Discrete DtN Map

Let us now consider the interface wise partition of the discrete DtN map as follows,

$$\mathbf{Z}^{(d)} = \begin{bmatrix} \mathbf{Z}_{11}^{(d)} & \mathbf{Z}_{12}^{(d)} & \cdots & \mathbf{Z}_{1n}^{(d)} \\ \mathbf{Z}_{21}^{(d)} & \mathbf{Z}_{22}^{(d)} & \cdots & \mathbf{Z}_{2n}^{(d)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Z}_{n1}^{(d)} & \mathbf{Z}_{n2}^{(d)} & \cdots & \mathbf{Z}_{nn}^{(d)} \end{bmatrix} \tag{3.6}$$

where n denotes the total number of interfaces in domain d. The discrete DtN map is dense, as it represents all-to-all interactions between unknowns residing at the interfaces, but this may be deceiving because as we will reveal in the sequel its non-block diagonal interactions have a low-rank structure. To see that lets consider the 2D decomposition in Fig. 3.1, that shows 5 air domains of $\lambda \times \lambda \times \lambda$ size discretized with $\lambda/10$ tetrahedron mesh at 50MHz. The center domain has 4 neighboring domains thus 4 interfaces, i.e. $n = 4$. This example although simple, it leads to indicative conclusions that hold for many more complicated situations encountered in a real decomposition. We now take this example to find out the singular values decay of each sub-block matrix. This helps to determine the rank deficiency of a sub-block matrix. Note that the domain FEM matrix $\mathbf{A}_{(d)}$ arising from finite elements is symmetric and the left and right restriction matrices are transpose to each other in (3.5), hence $\mathbf{Z}^{(d)}$ is also a symmetric matrix. It is also observed that interface $1-4$ are interchangeable if the meshes on interfaces are identical. Non-conforming interface meshes will change the interactions but following the same trend. Therefore we only present the results for the sub-block matrices from the first row, i.e., $\mathbf{Z}_{1j}^{(d)}(j = 1, \cdots, 4)$.

In Fig. 3.2, the normalized singular values decay of matrix blocks $\mathbf{Z}_{1j}^{(d)}(j = 1, \cdots, 4)$ corresponding to self and touching and separated interface interactions is presented. It can be observed that the singular values of the self-term DtN operator $\mathbf{Z}_{11}$ decay at the slowest rate, while the singular values of $\mathbf{Z}_{13}$, the block corresponding

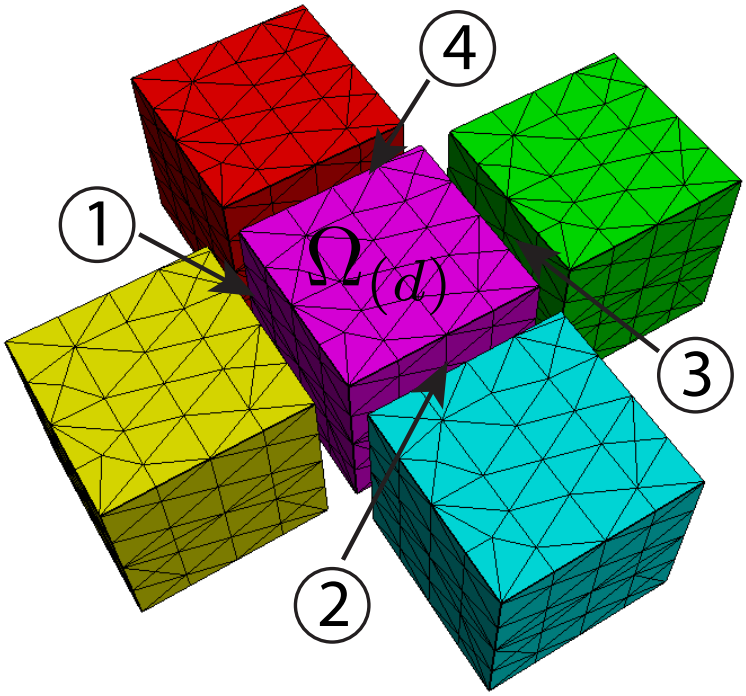


Figure 3.1. A computational model with 5 domains in an 2D decomposition topology used to demonstrate the rank-deficiency of some blocks of the discrete DtN map.

to separated interface interactions, decay at the fastest rate. The singular values of $\mathbf{Z}_{12}$ and $\mathbf{Z}_{14}$ decay at a moderate rate that is between the two extreme cases described above. This indicates that for errors of the order of $10^{-4} \sim 10^{-6}$, the numerical rank of $\mathbf{Z}_{13}$ is smaller than the matrix dimension, thus a rank-revealing range finder could potentially lead to computational savings.

To further verify this hypothesis of low numerical rank for separated interface interactions, another computational experiment is performed. In Fig. 3.3, a $\lambda \times \lambda \times L$ domain is used to compute the DtN map operator $\mathbf{Z}_{tx-rx}$ with varying separation L between the two interfaces. L ranges from 0.1 to 10 without changing total number of unknowns or mesh topology. For the purpose of comparison, the geometric tetrahedral mesh is morphed using the technique proposed in [68]. Fig. 3.4 shows the singular values decay of computed $\mathbf{Z}_{tx-rx}(L)$. As the Tx-to-Rx interface separation increases,

the decay of the singular values of the respective DtN block becomes more rapid, suggesting lower numerical ranks for larger separations.

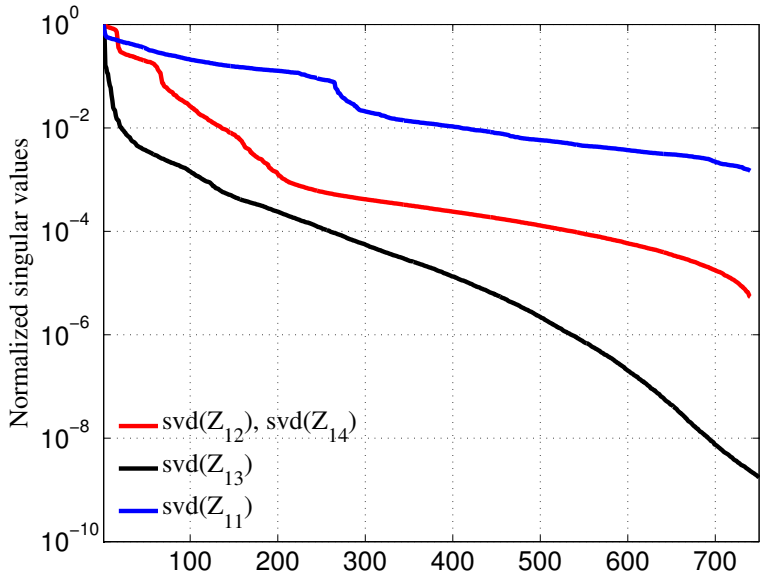


Figure 3.2. Decay of singular values of z-matrix of different interface combination.

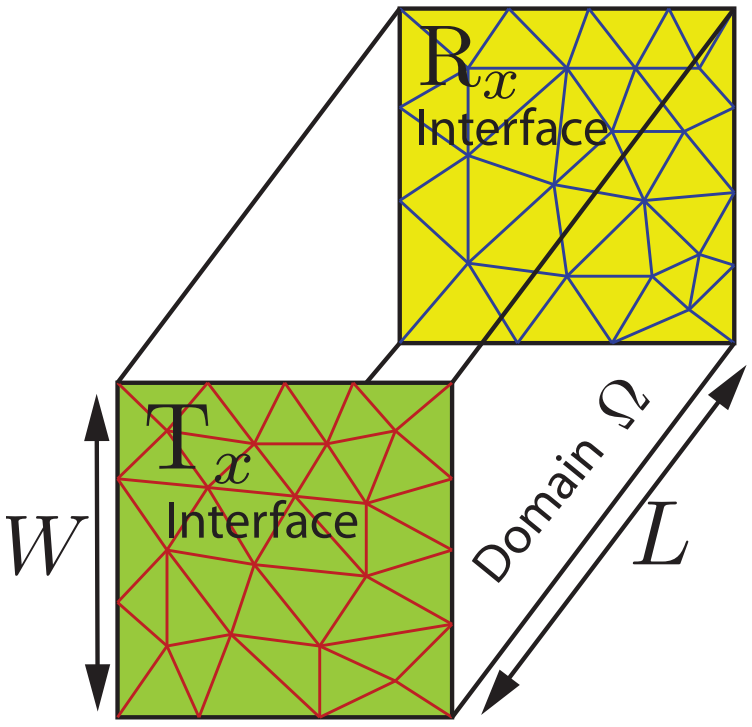


Figure 3.3. Domain setup with varying size for DtN map operator computation.

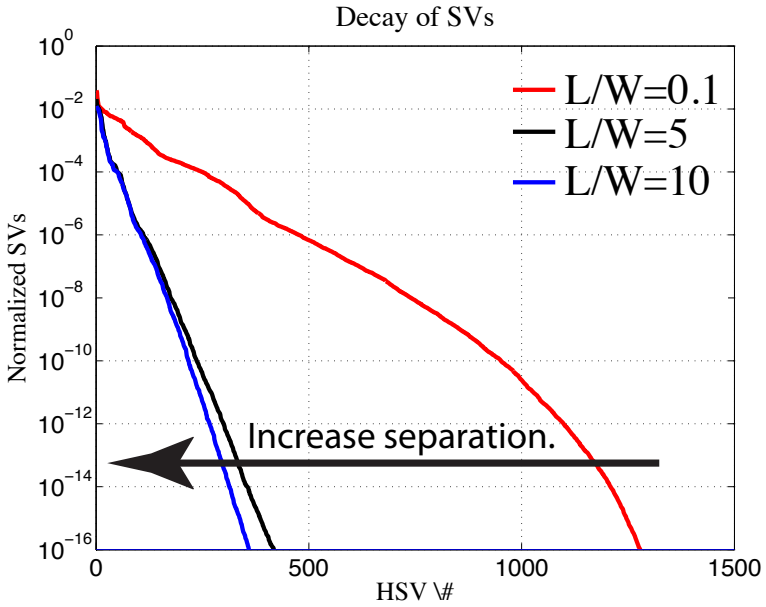


Figure 3.4. Decay of singular values of DtN map matrix of two opposite interfaces with varying separation.

The situation described in the above mentioned computational experiments is analogous to the block-wise (or group wise) rank deficiency for the impedance matrix in MoM when the matrix interactions represent well separated groups of unknowns [23].

With these observations in mind, the task now shifts at finding computationally efficient and reliable algorithms that would automatically detect and leverage the rank deficiency of the separated DtN blocks. In the next section, this task is accomplished by adopting ideas borrowed from the field of randomized computations.

3.3 DtN Map Operator Approximation via Randomized Computations

Assume DtN matrix block $\mathbf{Z}$ is numerically rank deficient, then it can be represented in the form,

$$\mathbf{Z} \approx \mathbf{U}\,\mathbf{V}, \tag{3.7}$$

where $\mathbf{Z} \in \mathbb{C}^{m\times n}$, $\mathbf{U} \in \mathbb{C}^{m\times k}$ and $\mathbf{V} \in \mathbb{C}^{k\times n}$ where $k \ll \min(m,n)$. This can be achieved by constructing an approximate set of orthonormal basis ($\mathbf{Q}$) for the range of the input matrix $\mathbf{Z}$, such that

$$\mathbf{Z} \approx \mathbf{Q}\mathbf{Q}^*\mathbf{Z}. \tag{3.8}$$

Therefore,

$$\mathbf{U} = \mathbf{Q}, \tag{3.9}$$

$$\mathbf{V} = \mathbf{Q}^*\mathbf{Z}. \tag{3.10}$$

The matrix $\mathbf{Q}$ needs to have as few columns as possible so that the original matrix can be represented in a much more compressed form. Therefore, from (3.5) and (3.6), for each non-diagonal sub-block matrix($i \neq j$), the approximated DtN matrix $\tilde{\mathbf{Z}}_{ij}^{(d)}$ can be written in the following low-rank decomposition form,

$$\tilde{\mathbf{Z}}_{ij}^{(d)} = \mathbf{Q}\mathbf{Q}^*(\mathbf{R}_i^e\mathbf{A}_{(d)}^{-1}\mathbf{R}_j^{eT}), \tag{3.11}$$

where $\mathbf{Q} \in \mathbb{C}^{m\times k}$ is a rank-k matrix, $\mathbf{A}_{(d)}$ is the domain finite element matrix, $\mathbf{R}_i^e$ and $\mathbf{R}_j^e$ are the binary restriction matrices mapping the vector of primal DoFs of domain d to the primal DoFs defined on interface i and j. Eq. (3.11) can be further written as,

$$\tilde{\mathbf{Z}}_{ij}^{(d)} = \mathbf{Q}(\mathbf{A}_{(d)}^{-*}(\mathbf{R}_i^{eT}\mathbf{Q}))^*\mathbf{R}_j^{eT}. \tag{3.12}$$

To be consistent with (3.7), the above formula is rewritten as

$$\tilde{\mathbf{Z}}_{ij}^{(d)} = \mathbf{Q}(\mathbf{R}_j^e\mathbf{X})^*, \tag{3.13}$$

therefore $\mathbf{U} = \mathbf{Q}$, $\mathbf{V} = (\mathbf{R}_j^e \mathbf{X})^*$, and $\mathbf{X}$ can be computed by solving following linear matrix equation,

$$\mathbf{A}_{(d)}^* \mathbf{X} = \mathbf{R}_i^{eT} \mathbf{Q}. \tag{3.14}$$

It is worth pointing out that to solve (3.14), only the direct factorization of $\mathbf{A}_{(d)}^*$ is necessary. This is due to the fact that in FETI-2λ, the domain matrix for reciprocal media is a symmetric matrix, therefore solving such linear equation is equivalent to solve

$$\mathbf{A}_{(d)} \overline{\mathbf{X}} = \overline{\mathbf{R}_i^{eT} \mathbf{Q}} \tag{3.15}$$

which only needs forward-backward substitution as the factorization of $\mathbf{A}_{(d)}$ that is readily available from the setup stage.

Up to this point, it is clear that the challenge becomes how to efficiently find a good low-rank approximation matrix $\mathbf{Q}$. A straightforward approach of finding the matrix $\mathbf{Q}$ is to use the k dominant left singular vectors of $\mathbf{A}_{(d)}$. However this becomes computationally prohibitive when the discretized domain size increases. In the next sections, we propose two numerically efficient algorithms to tackle this challenge.

3.3.1 Fixed Rank Randomized Algorithm

This section proposes a fixed rank randomized algorithm to find the $\mathbf{Q}$ matrix mentioned in above. It is important to note that the proposed algorithms in this and next sections share the same spirit of the randomized algorithms found in [5], however the most significant difference between the proposed algorithm and the work by Halko et. al. is that the DtN matrix is not readily available in our applications, the DtN matrix is computed on the fly via carefully incorporating direct factorization methods.

In essance, the goal of constructing the matrix $\mathbf{Q}$ is to be able to capture the most characteristics of the matrix $\mathbf{Z}$, such that the approximation in (3.8) holds. Before introducing the proposed algorithm, we define following parameters:

l: estimated required rank of a random matrix.

p: an over-sampling parameter

and $k = l + p$. The reason we need to define two different rank size parameters is because the rank size of $\mathbf{Q}$ that can provide a good approximation of $\mathbf{Z}$ is usually unknown for a given domain, therefore an over-sampling parameter p is adopted to provide some flexibility that is important for the accuracy of the proposed algorithm. The fixed rank randomized algorithm is proposed in Algorithm 1.

Algorithm 1 : Fixed-rank randomized computation of non-diagonal DtN

INPUT:
$\mathbf{A}_d \in \mathbb{C}^{N\times N}$: Domain FEM matrix.
l: Estimated column dimension of random matrix.
p: Number of over samples.

OUTPUT:
$\tilde{\mathbf{Z}}^{(d)}_{\mathrm{ij}} \in \mathbb{C}^{m\times n}$: Approximated off-diagonal sub-block of domain DtN matrix $\mathbf{Z}^{(d)}$.

DEFINITIONS:
$\mathbf{\Omega} \in \mathbb{C}^{n\times k}$: random matrix, where $k = l + p$.
$\mathbf{R}^e_i \in \mathbb{R}^{m\times N}, \mathbf{R}^e_j \in \mathbb{R}^{n\times N}$: restriction matrix mapping primal DoFs on interface i and j to volumetric primal DoFs of domain d.

1: `Generate a` $n \times k$ `random matrix` $\mathbf{\Omega}$.
2: `Solve` $\mathbf{A}_d\mathbf{Y} = \mathbf{R}_j^{eT}\mathbf{\Omega}$.
3: `Restrict` $\mathbf{Y}$ `via` $\mathbf{Y}' = \mathbf{R}_i^e\mathbf{Y}$.
4: `Compute` $\mathbf{Q}$ `through QR factorization` $\mathbf{Y}' = \mathbf{QR}$.
5: `Solve` $\mathbf{X}$ `through` $\mathbf{A}_d^*\mathbf{X} = \mathbf{R}_i^{eT}\mathbf{Q}$.
6: $\tilde{\mathbf{Z}}^{(d)}_{ij} = \mathbf{Q}(\mathbf{R}_j^e\mathbf{X})^*$.

Remark

1. Algorithm 1 uses a random matrix $\mathbf{\Omega}$ from the standard Gaussian distribution. Each entry of the random matrix is an independent Gaussian random variable with mean zero and variance one. The random matrix helps to find the required space of matrix $\mathbf{Y}$ as much as possible, therefore a random number generator with good quality plays an important role. In this work, the Marsenne Twister Pseudo-random Number Generator (MT-PRNG) [69] is adopted.

2. Step 2 involves solving the linear matrix equation that requires a matrix factorization. It is solved via $\mathbf{LDL}^T$ factorization that is available from the set-up stage of FETI-2λ, followed by forward backward substitution.

3. It is practically difficult to provide a good estimation of the required rank for the random matrix Ω, as the information of the domain DtN sub-block $\mathbf{Z}_{ij}^{(d)}$ is rarely known in advance. Various numerical experiments suggest $l = (0.1 \sim 0.2) \times \min(m, n)$ can provide a good approximation for general problems.

4. The usage of an over-sampling parameter p provides better flexibility while choosing the number of samples (equals to the number of column dimension) for the random matrix Ω. Empirical experiments suggest that $p = 5 \sim 10$ usually gives a good enough approximation.

3.3.2 Adaptive Randomized Algorithm

A disadvantage of the fixed-rank randomized algorithm proposed in Algorithm 1 is that it lacks a reliable way to recognize the minimized rank of the random matrix $\mathbf{\Omega}$ in order to provide a good approximation of the DtN map operator. In this section, we naturally extend the fixed-rank randomized algorithm to an adaptive implementation similar to [5]. In the proposed adaptive algorithm, the rank of the random matrix Ω is increased until the residual reaches a user-defined error tolerence ε.

In order to determine how well the basis matrix $\mathbf{Q}$ captures the dominant singular values of $\mathbf{Z}_{ij}^{(d)}$, we need to develop a reliable and efficient error estimator. The error estimator needs to be computed very efficiently to minimize the computational overhead. Ideally, the error can be approximated by computing

$$e = \left\| (\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{Z}_{ij}^{(d)} \right\|. \tag{3.16}$$

where $\|\cdot\|$ denotes the ℓ_2 operator norm. The error estimator in (3.16) involves matrix-matrix product which is inefficient in high performance computation. Instead, we use

the following error estimator,

$$e = \|(\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{y}\|, \tag{3.17}$$

where $\mathbf{y}$ is a standard Gaussian vector. The error esimator in (3.17) is much more efficient as the matrix-matrix product $\mathbf{Q}\mathbf{Q}^*$ is never explicitly formed, only matrix-vector product is involved. The error estimator in (3.17) is valid based on the following lemma borrowed from [5, 6],

Lemma 3.3.1 *Suppose* $\mathbf{A}$ *be a complex* $m \times n$ *matrix, for a given positive integer* r*, a real number* $\alpha > 1$*, and a set of independent standard Gaussian vectors* $\mathbf{w}_i, i = 1, 2, \cdots, r$*, then,*

$$\|\mathbf{A}\| \leq 0.8 \times \alpha \max_{i=1,2,\cdots,r} \|\mathbf{A}\mathbf{w}_i\|$$

except with probability α^{-r}*.*

The importance of the error estimator in (3.17) is that it converts the matrix-matrix operation into matrix-vector operation which significantly improves the computational efficiency by adopting a more loose estimation. The complete adaptive version of finding rank deficient DtN map operator is given in Algorithm 2.

Algorithm 2 takes the same matrix input with two extra parameters. One is an incremental number of samples for the random matrix termed as k, and the other parameter ε is an error tolerence that is used as an adaptive pass stopping criteria. The choice of these two parameters, that will be discussed later, becomes critical as they represent a trade-off between the algorithm's accuracy and efficiency.

Remark

1. The choice of k is worth some discussion. A small value of k makes the computation of QR factorization in step 7 and error estimation in step 8 more efficient, however the overall efficiency can be harmed by requiring more passes to reach a desired error

Algorithm 2 : Adaptive randomized computation of non-diagonal sub-block DtN

INPUT:
$\mathbf{A}_d \in \mathbb{C}^{N\times N}$: Domain FEM matrix.
k: Incremental column size of random matrix
ε: Error tolerance

OUTPUT:
$\tilde{\mathbf{Z}}_{\mathrm{ij}}^{(d)} \in \mathbb{C}^{m\times n}\,(i \neq j)$: Approximated off-diagonal sub-block of domain DtN matrix $\mathbf{Z}^{(d)}$.

DEFINITIONS:
$\|\mathbf{e}_p\|$: Norm of pth column vector of the error estimator matrix $\mathbf{e}$.
$\mathbf{\Omega} \in \mathbb{C}^{n\times k}(k \ll n)$: random matrix.
$\mathbf{R}_i^e \in \mathbb{R}^{m\times N}, \mathbf{R}_j^e \in \mathbb{R}^{n\times N}$: restriction matrix mapping primal DoFs on interface i and j to volumetric primal DoFs of domain d.

1: $q = 0$
2: **while** $\max\{\|\mathbf{e}_p\|, p = 1, 2, \cdots\} > \varepsilon$ **do**
3: $q \leftarrow q + 1$
4: Generate a $n \times k$ random matrix $\mathbf{\Omega}$.
5: Solve $\mathbf{A}\mathbf{Y}^q = \mathbf{R}_j^{eT}\mathbf{\Omega}$ and restrict $\mathbf{Y}^q$ via $\mathbf{y}^q = \mathbf{R}_i^e\,\mathbf{Y}^q$.
6: $\mathbf{y} \leftarrow [\mathbf{y}\,\mathbf{y}^q]$
7: Compute $\mathbf{Q}$ through QR factorization $\mathbf{y} = \mathbf{Q}\,\mathbf{R}$.
8: Construct error estimator $\mathbf{e} = (\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\,\mathbf{y}$
9: **end while**
10: Solve $\mathbf{X}$ through $\mathbf{A}_d^*\mathbf{X} = \mathbf{R}_i^{eT}\,\mathbf{Q}$.
11: $\tilde{\mathbf{Z}}_{ij}^{(d)} = \mathbf{Q}(\mathbf{R}_j^e\mathbf{X})^*$.

tolerence. Opposite effects can be observed by choosing a considerable large k value, and it also makes the final rank of $\mathbf{Q}$ potentially large (over-sampled). After various numerical experiments, we suggest to choose $k = 32$ or $k = 64$. This eventually gives a good approximation of the optimal rank of the matrix $\mathbf{Q}$ with only a small number of over samples.

2. The error tolerence ε is important as it controls the approximation accuracy of the rank deficient DtN matrix. In the presented numerical results, an error tolerence $\varepsilon = 0.01$ is used.

3. The proposed adaptive algorithm repeatedly computes the QR factorization shown in step 7. Fortunately, the computation is relatively small because the dimension of

$\mathbf{y} \in \mathbb{C}^{m \times k}$ where m denotes the number of DoFs defined only at one domain interface, which is significantly smaller than the total number of DoFs of a domain.

4. As mentioned above, the k blocking can lead to extra samples at the final adaptive loop, such strategy is generally harmless as the proposed algorithm is accelerated by exploiting matrix-level linear algebra packages i.e. BLAS3 [70].

3.3.3 Complexity Analysis

Having established the algorithm formulation, it is appropriate to examine the computational complexity of the proposed scheme. In this section, only the complexity of Algorithm 2 is discussed. The complexity of Algorithm 1 is identical to Algorithm 2 as the only difference is that the rank of matrix $\mathbf{Q}$ is fixed in advance. The computational complexity of the proposed algorithm is stated in the following theorem.

Theorem 3.3.2 *The computational complexity of the adaptive randomized algorithm to compute non-diagonal sub-blocks of a domain DtN map operator matrix scales as* $\mathcal{O}(N^{4/3})$ *where* N *is the total unknowns of the computational domain.*

Proof The computational time of the proposed algorithm in Algorithm 2 can be decomposed into

$$t_{\mathbf{Z}_{ij}^{(d)}} = n_{\text{pass}} \times (t_{\text{rg}} + t_{\mathbf{Y}} + t_{\text{QR}} + t_{\text{error}}) + t_{\mathbf{X}} + t_{\text{m}\times\text{m}}, \tag{3.18}$$

where n_{pass} denotes the number of adaptive pass to reach the desired error tolerance, t_{rg} is the time to generate the random matrix, $t_{\mathbf{Y}}$ is the time to solve the linear equation in step 5, t_{QR} is QR factorization time in each adaptive pass, t_{error} is the time to compute the error estimator, $t_{\mathbf{X}}$ is the time to solve the linear equation after $\mathbf{Q}$ is generated, and $t_{\text{m}\times\text{m}}$ is the matrix-matrix product time for computing the non-

diagonal DtN matrix. The time complexity for each component is discussed below. Note that in the complexity analysis below, we define the following notations.

N : total unknowns of the computational domain.

D : number of decomposed domains.

n_d : unknown number of domain d, $n_d \approx N/D$.

$n_{\mathcal{I}}$: unknown number on a domain interface $\mathcal{I}$, $n_{\mathcal{I}} = \alpha n_d$ where α is a scalor.

(1) t_{rg}: For a given random matrix $\mathbf{\Omega} \in \mathbb{C}^{n_{\mathcal{I}} \times k}$, each entry is an independent random number based on the Gaussian distribution, therefore the random matrix generation time ought to scale linearly with the number of matrix entry, i.e., $t_{\text{rg}} \sim \mathcal{O}(n_{\mathcal{I}} k)$. Since k is constant, $t_{\text{rg}} \sim \mathcal{O}(n_{\mathcal{I}})$.

(2) $t_{\mathbf{Y}}$: Due to the fact that the factorization of domain FEM matrix $\mathbf{A}$ is readily available, solving linear equation in step 5 only involves forward and backward substitution, therefore $t_{\mathbf{Y}} \sim \mathcal{O}(n_d^{4/3})$ due to the fill-in in $\mathbf{L}$ factor. Note this is true for 3D problems re-ordered with nested dissections i.e. MeTiS.

(3) t_{QR}: It is well known that the QR factorization can be constructed by using Gram–Schmidt process, which has a numerical complexity $\mathcal{O}(m)$, where m is the row dimension of matrix $\mathbf{y}$ in step 6. Therefore, $t_{\text{QR}} \sim \mathcal{O}(n_{\mathcal{I}})$.

(4) t_{error}: The proposed error estimator in step 8 has a complexity of $\mathcal{O}(mk)$, since k is constant, $t_{\text{error}} \sim \mathcal{O}(n_{\mathcal{I}})$.

(5) $t_{\mathbf{X}}$: It is aforementioned that to solve the linear equation in step 10, one can reuse the factorization of domain FEM matrix $\mathbf{A}_d$ instead of factorizing $\mathbf{A}_d^*$, this is due to the fact that $\mathbf{A}_d$ is symmetric. Thus, step 10 only involves matrix forward and backward substitution, $t_{\mathbf{X}} \sim \mathcal{O}(n_d^{4/3})$.

(6) $t_{\mathrm{m}\times\mathrm{m}}$: The complexity of matrix product $\mathbf{R}_j^e\mathbf{X}$ is $\mathcal{O}(nkn_{\text{pass}})$, and the complexity of left multiplying with $\mathbf{Q}$ is $\mathcal{O}(mkn_{\text{pass}})$, therefore the total complexity is $\mathcal{O}((m+n)kn_{\text{pass}})$, since k is constant and n_{pass} is proportional to the rank of the matrix block that is proportional to the size of the interface, i.e. $\mathcal{O}((n_d^{2/3})^2)$, thus $t_{\mathrm{m}\times\mathrm{m}} \sim \mathcal{O}(m+n) = \mathcal{O}(n_d^{4/3})$.

Note that D is constant for a specific domain decomposition, and α is a scalar, therefore $\mathcal{O}(n_{\mathcal{I}}) = \mathcal{O}(n_d) = \mathcal{O}(N)$. To summarize, $t_{\mathbf{Z}_{ij}^{(d)}} \sim \mathcal{O}(N)$. ∎

3.3.4 Error Analysis

In this section, we aim to analyze and quantify the accuracy of the approximated DtN matrix. For a given non-diagonal sub-block DtN matrix $\mathbf{Z}_{ij}^{(d)} \in \mathbb{C}^{m\times n}$, the error residual can be defined as

$$\mathbf{e} = (\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\mathbf{Z}_{ij}^{(d)}, \tag{3.19}$$

where $\mathbf{Q} \in \mathbb{C}^{m\times(k+p)}, (k+p) \ll \min(m,n)$ is an orthonormal matrix. Based on the major results of [5, 7], the probability of the ℓ_2-norm of the true error satisfies

$$\|\mathbf{e}\| \leq \left[1 + 9\sqrt{k+p}\cdot\sqrt{\min(m,n)}\right]\sigma_{k+1}, \tag{3.20}$$

is at least $1 - 3\cdot p^{-p}$, where σ_{k+1} is the $k+1$ largest singular value of $\mathbf{Z}_{ij}^{(d)}$. A rigorous proof of (3.20) can be found in [5]. Two important observations arising from this error bound are discussed as below.

Firstly, the error bound in (3.20) justifies that the proposed algorithm works best for rank deficient input matrices with fast decaying singular values, since one can achieve a very small upper error limit σ with a considerably small k. This eventually leads to effcieintly generating the random matrix and solving the resulting linear matrix equation. The other observation is that the error bound in (3.20) also justifies the use of a small over sample parameter like $p = 5$. In fact, for rank deficient DtN

matrices with well fast decaying singular values, $p = 3$ usually can provide a decent approximation.

3.4 Domain Discrete DtN Map Operator Computation

In the previous section, an accurate and efficient randomized algorithm to compute a single off-diagonal DtN map operator matrix in (3.6) was proposed. In this section, we discuss the overall computational framework to compute the domain DtN map operator arised in FETI-2λ DDM. We first discuss the computation of self term DtN map operators, i.e., the diagonal sub-block matrices in (3.6), then discuss an efficient implementation to compute non-self term DtN blocks based on algorithms to compute a single off-diagonal sub-block matrix. Finally, two strategies to further improve the computational accuracy and efficiency of the proposed framework are discussed.

3.4.1 Self term DtN Map Operator

In Fig. 3.2, it is shown that when $i = j$, the self term DtN block $\mathbf{Z}_{ij}^{(d)}$ is not rank deficient. In these cases, the proposed randomized algorithm does not fit such computation as the singular values decay very slowly. A direct computation approach with matrix selective inverse direct solver will be adopted. A detailed discussion of such selective inverse computation method is available in [71]. For the sake of completeness, we briefly discuss the core ideas.

Let us define z_{ij} as the (i, j) entry of matrix $\mathbf{Z}^{(d)}$, and we assume the LU factorization of its corresponding domain FEM matrix $\mathbf{A} = \mathbf{LU}$ is readily available, then the direct approach to find z_{ij} is formulated as:

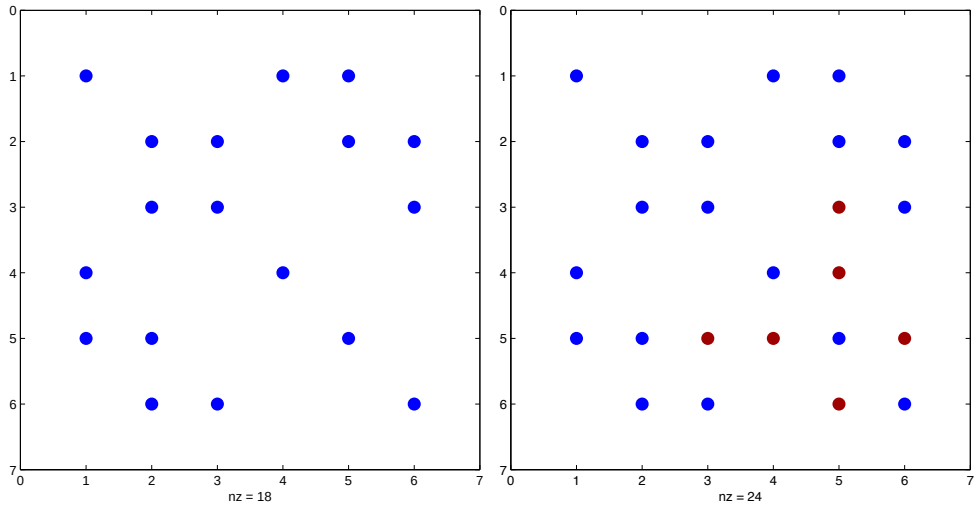
1. `Find` j`th column of` $\mathbf{Z}^{(d)}$ `through` $\mathbf{z}_{*j} = (\mathbf{LU})^{-1}\mathbf{e}_j = \mathbf{U}^{-1}(\mathbf{L}^{-1}\mathbf{e}_j)$.

1.1 `Solve` $\mathbf{x}$ `from` $\mathbf{Lx} = \mathbf{e}_j$.

1.2 `Solve` $\mathbf{z}_{*j}$ `from` $\mathbf{Uz}_{*j} = \mathbf{x}$.

2. $z_{ij} = (\mathbf{z}_{*j})_i$.

The above $\mathbf{e}_j$ denotes a zero vector except the jth entry is one, $\mathbf{z}_{*j}$ represents the jth column vector of $\mathbf{Z}^{(d)}$, and operator $(\cdot)_i$ represents the ith entry of the vector. Although the approach is straightforward, the challenge lies in the fact that the $\mathbf{U}$ and $\mathbf{L}$ factors of a sparse matrix are usually much dense than the sparse matrix itself. Fig. 3.5 shows the sparsity pattern comparison of a symmetric sparse matrix and its LU factors. Thus the direct approach requires intensive computational resource as the entire $\mathbf{U}$ and $\mathbf{L}$ matrix need to be loaded into memory, and specifically, the efficiency is harmed significantly when only a selective set of entries of the inverse matrix are interested. The key idea is to utilize partial $\mathbf{L}$ and $\mathbf{U}$ by exploiting the sparsity pattern of the original matrix and blocks of interest, this goal can be achieved by the contruction of an elimination tree of the domain FEM matrix.



(a) (b)

Figure 3.5. A structurally symmetric matrix with the factorization. (a) Sparsity pattern of the matrix; (b) Sparsity pattern of the factorization, red dots indicate fill-in factor values.

To create the elimination tree, we start from the matrix sparsity pattern. Fig. 3.6(a) shows a non-directed graph $G(\mathbf{A}) = (V, E)$ based on the sparsity pattern of $\mathbf{L} + \mathbf{U}$ shown in Fig. 3.5(b), where V is a set of vertex with each one representing

one column, and E is a set of edge with each one representing an matrix entry. Thus an elimination tree is defined as following,

Definition The elimination tree of $\mathbf{A} = \mathbf{LU}$ is a tree of N nodes, with the ith node corresponding to the ith column of $\mathbf{L}$, and its parent is found s.t.

$$\text{parent}(i)=\min\{j : j > i \text{ and } \ell_{ij} \neq 0, \text{ where } \ell_{ij} \text{ is an entry of } \mathbf{L}\}$$

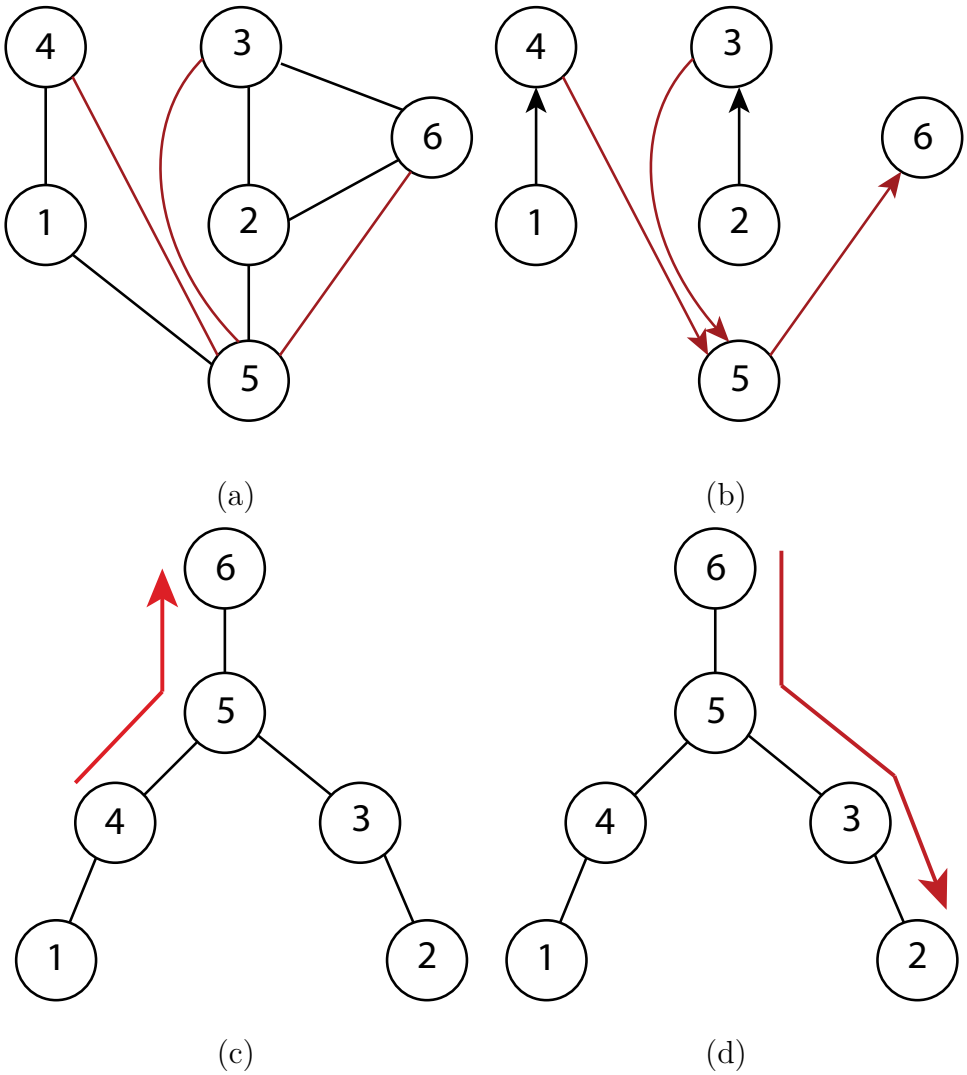


Figure 3.6. Non-directed graph representing the sparsity pattern of matrix factors and corresponding elimination tree, red edges represent added fill-in entries; (a) The graph of the original matrix factors $\mathbf{L} + \mathbf{U}$; (b) The corresponding elimination tree; (c-d) Traversal path of the elimination tree to find z_{24}.

An example of the elimination tree following the definition is constructed in Fig. 3.6(b). The elimination tree helps to solve selective inverse entry via loading the appropriate **L** and **U** factors based on the theorem [67] below,

Theorem 3.4.1 *To compute a particular entry z_{ij} in* **Z***, the only factors have to be loaded are the* **L** *factors on the elimination tree path from node j up to the root node, and the* **U** *factors on the path going back from the root to node i.*

The theorem states that the computational cost of calculating a single entry is related to the computational effort of tree traversal from jth node to the root, and then from the root to the ith node. An example traversal path is given in Fig. 3.6(c)-(d) in order to find entry z_{24}. The benefit of such tree traversal is that it avoids to solve z_{44} as it has to in a traditional forward-backward substitution approach.

In this work, the inverse entries of diagonal blocks are computed directly. In such case, the above technique is implemented in a block fashion. The right-hand-side (RHS) vectors (**e**) are re-partitioned such that the same leaves of the elimination tree are accessed at minimized times, leading to a minimized factor matrix blocks into the memory. Reordering techniques such as post-order partitioning (PoP) and Hypergraph Partitioning (HP) [71] exist to further improve the efficiency of the selective inverse computation. These algorithms are implemented and readily available in the MuMPS linear algebra package [67] that is adopted in this work to solve non-rank deficient diagonal DtN matrix blocks.

3.4.2 Non-self term DtN Map Operator

Having established the algorithm to compute a single non-diagonal DtN matrix block in previous sections, now we propose the computational scheme to compute the entire domain DtN matrix. Assume domain d has n domain interfaces, in order to compute all the non-diagonal DtN block matrices in (3.6), the randomized algorithm

need to be executed $\frac{1}{2}(n+1)n$ times, this harms the total computational efficiency particularly when dealing with three-dimensional decomposition when n is larger.

The aforementioned issue can be somehow overcomed by consolidating multiple interfaces into a single one. Note that Algorithm 1-2 computes a single block $\mathbf{Z}_{ij}^{(d)}$ of $\mathbf{Z}^{(d)}$ which is mathematically defined as

$$\mathbf{Z}_{ij}^{(d)} = \mathbf{R}_i^e \mathbf{Z}^{(d)} \mathbf{R}_j^{eT} \tag{3.21}$$

where $\mathbf{R}_i^e$ and $\mathbf{R}_j^e$ denote the restriction mapping matrix of primal DoFs of domain d to the ones at interface $\mathcal{I}_i$ and $\mathcal{I}_j$ such that $\mathbf{e}_i^{(d)} = \mathbf{R}_i^e \mathbf{e}^{(d)}$, where the DoFs vector $\mathbf{e}_i^{(d)}$ is defined on the surface triangles list of interface $\mathcal{I}_i$. Algorithm 1-2 eventually computes a coupling matrix among the DoFs on two different interfaces. This observation enables to develope a more efficient approach to compute non-self DtN matrices. For a given interface $\mathcal{I}_i$, the non-self DtN matrices associated with $\mathcal{I}_i$ can be computed by consolidating all the domain interfaces $\mathcal{I}_j, j \neq i$ a shown below,

$$\begin{aligned}
\mathbf{Z}_{1*}^{(d)} &= \left[\mathbf{Z}_{12}^{(d)}\ \mathbf{Z}_{13}^{(d)}\ \cdots\ \mathbf{Z}_{1n}^{(d)}\right], \\
\mathbf{Z}_{2*}^{(d)} &= \left[\mathbf{Z}_{23}^{(d)}\ \cdots\ \mathbf{Z}_{2n}^{(d)}\right], \\
&\cdots \\
\mathbf{Z}_{(n-1)*}^{(d)} &= \left[\mathbf{Z}_{(n-1)n}^{(d)}\right],
\end{aligned} \tag{3.22}$$

where $\mathbf{Z}_{i*}^{(d)}$ denotes the DtN matrix representing DoFs coupling between interface $\mathcal{I}_i$ and a virtual consolidated interface $\mathcal{I}_* = \{\bigcup \mathcal{I}_j : j = i+1, i+2, \cdots, n\}$. The new partition of the DtN matrix is given in (3.30). The new block partition enables one only needs to execute the randomized algorithm once for all the $\mathbf{Z}_{ij}^{(d)}, j \neq i$. Algorithm 3 shows the proposed approach.

$$\mathbf{Z}^{(d)} = \begin{bmatrix} \mathbf{Z}_{11}^{(d)} & \mathbf{Z}_{12}^{(d)} & \mathbf{Z}_{13}^{(d)} & \cdots & \mathbf{Z}_{1n}^{(d)} \\ \mathbf{Z}_{21}^{(d)} & \mathbf{Z}_{22}^{(d)} & \mathbf{Z}_{23}^{(d)} & \cdots & \mathbf{Z}_{2n}^{(d)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{Z}_{(n-1)1}^{(d)} & \mathbf{Z}_{(n-1)2}^{(d)} & \mathbf{Z}_{(n-1)3}^{(d)} & \cdots & \mathbf{Z}_{(n-1)n}^{(d)} \\ \mathbf{Z}_{n1}^{(d)} & \mathbf{Z}_{n2}^{(d)} & \mathbf{Z}_{n3}^{(d)} & \cdots & \mathbf{Z}_{nn}^{(d)} \end{bmatrix} \begin{matrix} \\ \rightarrow \\ \rightarrow \\ \\ \rightarrow \\ \\ \end{matrix} \begin{matrix} \mathbf{Z}_{1*}^{(d)} \\ \mathbf{Z}_{2*}^{(d)} \\ \vdots \\ \mathbf{Z}_{(n-1)*}^{(d)} \\ \\ \end{matrix} \tag{3.23}$$

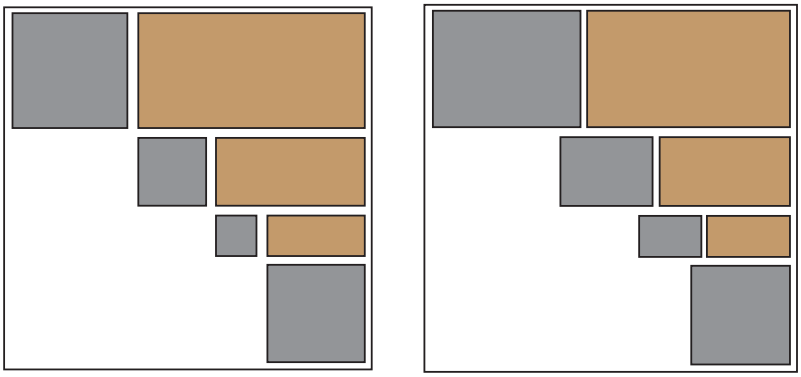
Remark

1. After consolidating the interfaces, the resulting DtN matrix $\mathbf{Z}_{i*}^{(d)} \in \mathbb{C}^{n_{\mathcal{I}_i} \times n_{\mathcal{I}_*}}$ becomes more skinny as the new column dimension is

$$n_{\mathcal{I}_*} = \sum_{j=i+1}^{n} n_{\mathcal{I}_j} \tag{3.24}$$

however the row dimension remains as the same, therefore the numerical rank k to approximate the new DtN matrix has the same upper bound as $k < n_{\mathcal{I}_i}$. Although it should be realized that more sample vectors are needed in order to achieve the same accuracy of the same row DtN matrix approximation.

2. It is observed that such consolidation strategy usually leads to a higher rank because some interactions are touching i.e. domain edge (interface-to-interface). In this work, a buffer region surrounding the domain edge with a user defined radius r_{buffer} is incorporated to avoid the issue. The $\mathbf{Z}$ entries corresponding to those DoFs defined within the buffer region are computed with a direct selective inverse approach. Effectively the partition shown in (3.30) is modified. A comparison of $\mathbf{Z}$ partition with and without buffer is shown in Fig. 3.7.



(a) (b)

Figure 3.7. The comparison of **Z** partition with and without buffer regions; (a) Without buffer regions; (b) With buffer regions.

3.4.3 Complexity Analysis

Since the algorithms to compute self and non-self DtN matrices have been proposed, we can present the total computational complexity to compute a domain DtN map operator, which is stated in the following theorem.

Theorem 3.4.2 *Given the matrix factorization of a domain d, the computational complexity of computing the domain DtN map operator matrix scales as $\mathcal{O}(n^{4/3})$ where n is the unknown size of the computational domain.*

Proof The computational time of the proposed scheme to compute a domain DtN map operator can be decomposed into

$$t_d = t_{\text{self}} + t_{\text{non-self}} \tag{3.25}$$

where t_{self} represents the time to compute its diagonal DtN blocks and $t_{\text{non-self}}$ being the time to compute its non-diagonal blocks. In Theorem 3.3.2, we have shown that $t_{\text{non-self}} \sim \mathcal{O}(N)$. Here, we only need to investigate t_{self}.

Since the domain matrix factorization is readily available, the matrix selective inverse algorithm presented above only needs to compute the entries by performing

Algorithm 3 : Adaptive randomized computation of non-diagonal DtN

INPUT:
$\mathbf{A}_d \in \mathbb{C}^{N\times N}$: Domain FEM matrix.
k: Incremental column size of random matrix
ε: Error tolerance

OUTPUT:
$\{\tilde{\mathbf{Z}}_{i*}^{(d)}\} \in \mathbb{C}^{n_{\mathcal{I}_i}\times n_{\mathcal{I}_*}}$ $(i = 1, 2, \cdots, n_{\mathcal{F}}-1)$: Approximated off-diagonal sub-block of domain DtN matrix $\mathbf{Z}^{(d)}$.

DEFINITIONS:
$n_{\mathcal{F}}$: Number of domain interface.
$n_{\mathcal{I}}$: DoFs size on domain interface $\mathcal{I}$.
$\|\mathbf{e}_p\|$: Norm of pth column vector of the error estimator matrix $\mathbf{e}$.
$\mathbf{\Omega} \in \mathbb{C}^{n_{\mathcal{I}_*}\times k}$: random matrix.
$\mathbf{R}_i^e \in \mathbb{R}^{n_{\mathcal{I}_i}\times N}, \mathbf{R}_*^e \in \mathbb{R}^{n_{\mathcal{I}_*}\times N}$: restriction matrix mapping primal DoFs on interface $\mathcal{I}_i$ and $\mathcal{I}_*$ via consolidating surface triangles of interface $\mathcal{I}_j, j = i+1, i+2, \cdots, n_{\mathcal{F}}$ to volumetric primal DoFs of domain d.

for $i = 1, 2, \cdots, n_{\mathcal{F}} - 1$ **do**
  Construct $\mathcal{I}_* = \bigcup \mathcal{I}_j, j = i+1, i+2, \cdots, n_{\mathcal{F}}$
  q = 0
  while $\max\{\|\mathbf{e}_p\|, p = 1, 2, \cdots\} > \varepsilon$ **do**
    $q \leftarrow q + 1$
    Generate a $n_{\mathcal{I}_*} \times k$ random matrix $\mathbf{\Omega}$.
    Solve $\mathbf{A}\mathbf{Y}^q = \mathbf{R}_*^{eT}\mathbf{\Omega}$ and restrict $\mathbf{Y}^q$ via $\mathbf{y}^q = \mathbf{R}_i^e\,\mathbf{Y}^q$.
    $\mathbf{y} \leftarrow [\mathbf{y}\ \mathbf{y}^q]$
    Compute $\mathbf{Q}$ through QR factorization $\mathbf{y} = \mathbf{Q}\,\mathbf{R}$.
    Construct error estimator $\mathbf{e} = (\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\,\mathbf{y}$
  end while
  Solve $\mathbf{X}$ through $\mathbf{A}_d^*\mathbf{X} = \mathbf{R}_i^{eT}\,\mathbf{Q}$.
  $\tilde{\mathbf{Z}}_{i*}^{(d)} = \mathbf{Q}(\mathbf{R}_*^e\mathbf{X})^*$.
end for

forward and backward substitution with partial $\mathbf{L}$ and $\mathbf{U}$ factors. It is obvious that such operation is a $\mathcal{O}(n_d^{4/3})$ complexity task, where $n_d \approx N/D$, therefore, $t_{\text{self}} \sim \mathcal{O}(N^{4/3})$. The total time complexity for a given domain d is $t_d \sim \mathcal{O}(N^{4/3})$. ∎

Next, we analyzie the memory complexity of the proposed computational scheme. As a comparison, we first calculate the memory storage requirement of the direct method. For a domain d with $n_{\mathcal{F}}$ interfaces, the memory requirement of the direct selective inverse matrix method is

$$\alpha \times \left[n_{\mathcal{I}}^2 + n_{\mathcal{I}} \right], \tag{3.26}$$

where $n_{\mathcal{I}} = \sum_{i=1}^{n_{\mathcal{F}}} n_{\mathcal{I}_i}$ with $n_{\mathcal{I}_i}$ representing the unknowns at interface $\mathcal{I}_i$ and α is a scalar representing the byte size of an matrix entry, depending on desired floating-point computation precision. To calculate the memory storage requirement based on Algorithm 3. We assume the dimension of the range approximation matrix $\mathbf{Q}$ for each row partition is $n_{\mathcal{I}_i} \times k_i$, therefore to store $\tilde{\mathbf{Z}}_{i*}^{(d)}$, one needs to have $\alpha \times (n_{\mathcal{I}_i} \times k_i + k_i \times n_{\mathcal{I}_*})$ bytes of memory, where $n_{\mathcal{I}_*} = \sum_{j=i+1}^{n_{\mathcal{F}}} n_{\mathcal{I}_j}$. By taking account of the diagonal block DtN matrices, the total memory storage requirement for domain d with the proposed computational scheme is

$$\alpha \times \left[\frac{1}{2} \sum_{i=1}^{n_{\mathcal{F}}} (n_{\mathcal{I}_i}^2 + n_{\mathcal{I}_i}) + \sum_{i=1}^{n_{\mathcal{F}}-1} (k_i \sum_{j=i}^{n_{\mathcal{F}}} n_{\mathcal{I}_j}) \right]. \tag{3.27}$$

In order to compare (3.26) and (3.27), we make a reasonable assumption that each interface has the same number of surface DoFs, therefore $n_{\mathcal{I}_i} = n_{\mathcal{I}}/n_{\mathcal{F}} = \beta$. We also give a empircal guess for $k_i = 0.1 \times n_{\mathcal{I}_i}$. Then (3.26) becomes

$$\alpha \times (\frac{1}{2} n_{\mathcal{F}}^2 \beta^2 + \frac{1}{2} n_{\mathcal{F}} \beta), \tag{3.28}$$

and (3.27) can be approximated as

$$\alpha \times \left\{ \left[\frac{1}{2} n_{\mathcal{F}} + \frac{1}{20} (n_{\mathcal{F}}^2 + n_{\mathcal{F}} - 2) \right] \beta^2 + \frac{1}{2} n_{\mathcal{F}} \beta \right\} \tag{3.29}$$

It is observed that the memory usage of both methods is a quadratic function of β, i.e,. the number of unknowns at the domain interfaces. From (3.28) and (3.29), the first order term are identical, only the second order term are different which are plotted in Fig. 3.8. It can be concluded that the proposed scheme memory complexity scales almost linearly w.r.t. number of interfaces DoFs while the non-compressed approach scales quadratically.

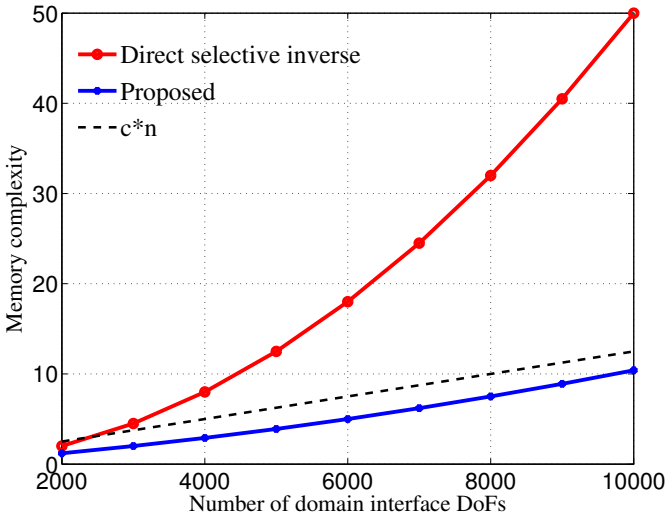


Figure 3.8. Memory complexity with respect to the number of domain interface DoFs.

3.4.4 Grouping

The scalability shown in Fig.3.8 suggests that the percentage of memory saving increases as a domain has more interfaces. For example, at $n_{\mathcal{F}} = 4$, the proposed scheme's memory saving is about 64% while at $n_{\mathcal{F}} = 10$, a 80% memory saving can be achieved. This means the proposed scheme saves more memory for center domains as these domains have more neighbors while the memory saving will relatively decreases for domains at the computational domain boundaries or corners. Based on this observation, in order to save more memory of a domain DtN, one can split a domain interface into multiple triangle groups. Assume a domain d has n neighbors and each domain interface is splitted into k groups, the decomposed discrete DtN map operator is shown in (3.30),

$$\mathbf{Z}^{(d)} = \begin{bmatrix} \mathbf{Z}^{(d)}_{1,1} & \mathbf{Z}^{(d)}_{1,2} & \cdots & \mathbf{Z}^{(d)}_{1,n\times k} \\ \mathbf{Z}^{(d)}_{2,1} & \mathbf{Z}^{(d)}_{2,2} & \cdots & \mathbf{Z}^{(d)}_{2,n\times k} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Z}^{(d)}_{n\times k,1} & \mathbf{Z}^{(d)}_{n\times k,2} & \cdots & \mathbf{Z}^{(d)}_{n\times k,n\times k}. \end{bmatrix} \tag{3.30}$$

Although using larger k leads to more memory saving, it is very possible the computation becomes more time consuming as it requires more repeated full solution of the diagonal blocks with the matrix selective inverse computation and randomized runs for the off-diagonal parts. To verify our hypothesis, a numerical experiment is conducted. Fig. 3.9 shows a scattering problem with a free-space air box due to an incident wave impinging at $\phi = \theta = 0^o$. The entire computational geometry is decomposed with 9 domains in a 3×3 2D decomposition. The computational time and memory statistics for the discrete DtN matrix of the center domain Ω_5 with different group number k at each interface is shown in Table 3.1.

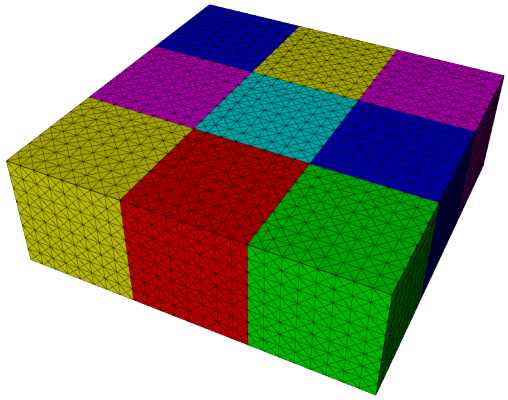


Figure 3.9. A free-space scattering problem decomposed into 9 domains in an structured 2D decomposition topology.

Table 3.1. Computational Statistics of the Discrete DtN for the Central Domain in a 2D Decomposition ($\varepsilon = 1.e-6$).

Method	k	Time (diagonal)	Time (off-diagonal)	Total Time	Total Memory [MB]
Direct	-	-	-	00:16:48	583.3
Proposed	1	00:06:42	00:08:11	00:14:53	254.9
	2	00:04:50	00:09:10	00:14:00	189.9
	3	00:03:28	00:12:40	00:15:41	139.9

It can be concluded that $k = 2$ provides a "sweat point" that both considerable time and memory savings are achieved. The parameter choice of $k = 2$ will be used throughout the simulations in this work unless explicitly stated otherwise.

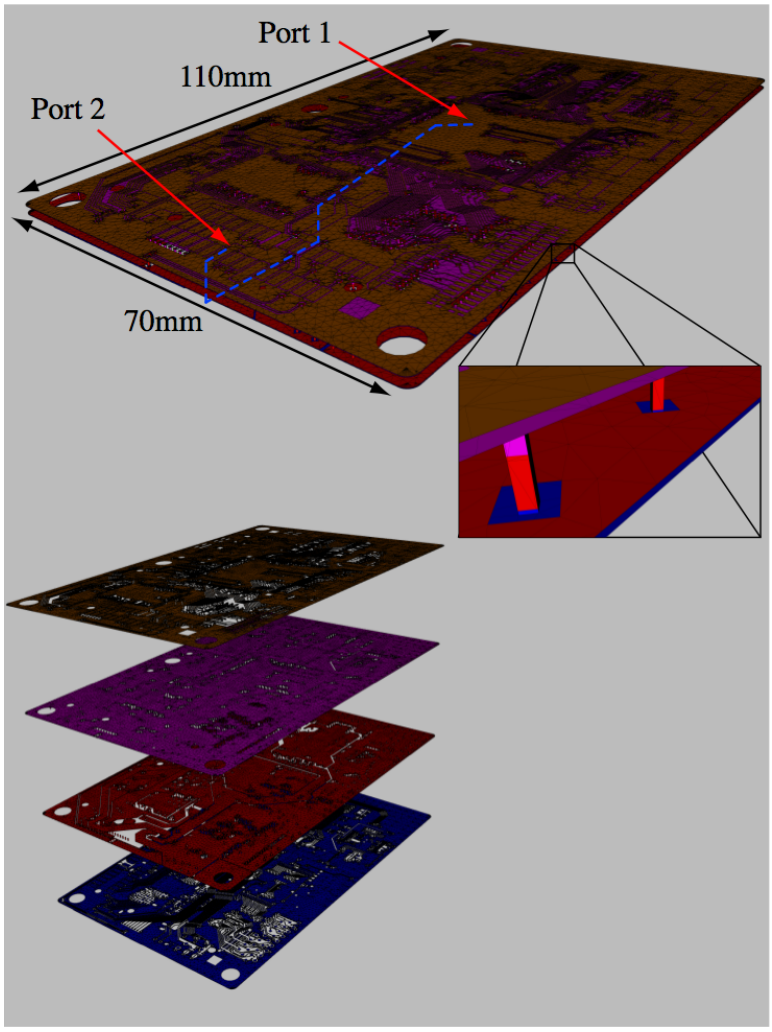
3.5 Numerical Results

In this section, two different numerical examples that illustrate the accuracy and efficiency of the proposed algorithms are presented. The computational codes were developed in C++ and compiled with Intel Compiler Suite 11.1 with -O3 performance optimization turned on. The adopted BLAS library is Intel Mathmatical Kernel Library (Intel MKL) [72]. For serial runs, the simulation was conducted on a Mac Pro with two 2.8GHz Intel Xeon quad-core processors and 32GB of RAM. For parallel runs, the numerical results were achieved using a in-house developed cluster of 10 Mac Pro with two 2.8GHz Intel Xeon quad-core processors and 32GB of RAM that are connected through a gigabyte ethernet network. The same computing environment was applied to all the numerical experiments throughout the manuscript unless explicitly stated otherwise.

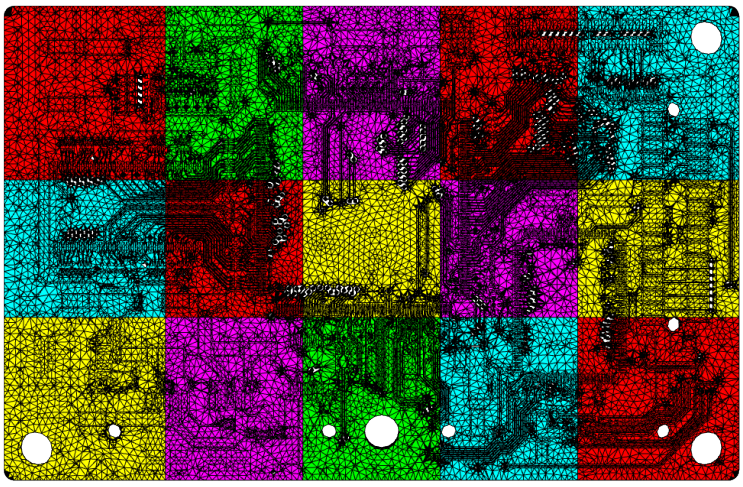
3.5.1 Printed Circuit Board

We first consider a realistic, commerical-graded computational problem that is the signal integrity (SI) analysis of a multi-layered printed circuit board (PCB) shown in Fig. 3.10(a). The PCB measures 110mm×70mm with about 5,000 vias placed between all four different layers. The entire computational domain is paritioned in a two dimensional structured decomposition with 15 domains as shown in Fig. 3.10(b), and discretized with a total number of 1,234,054 tetrahedrons, resulting 7,225,246 second-order FEM unknowns. In this simulation, port 1, at the center of the PCB (top layer), is excited and the signal is received on a second waveport placed at the middle layer of the PCB structure. The signal trace spans three layers and half of the PCB length, as shown with blue dashed line in Fig. 3.10(a).

Fig. 3.11 shows the computational time and required memory for computing each z-matrix in each domain of the decomposition. In both figures, the far left bar represents the time and memory usage of the direct z-matrix computation method, and



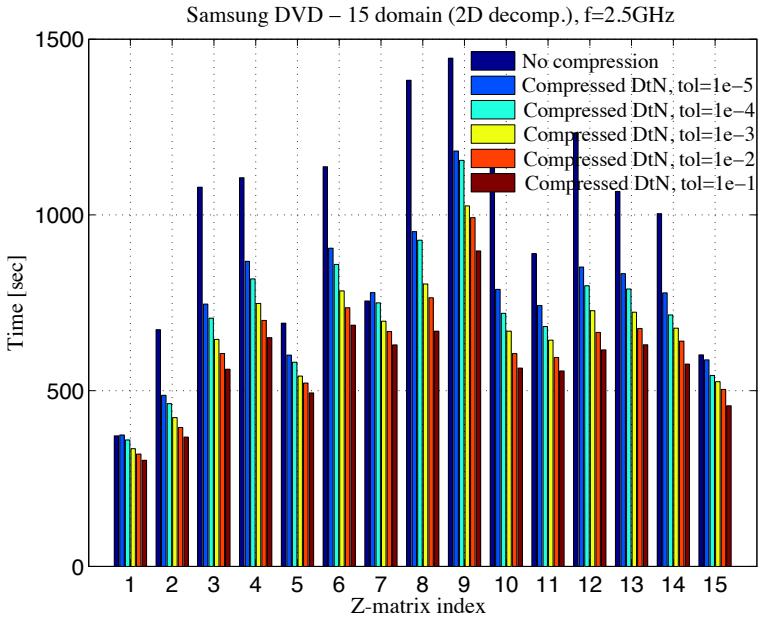
(a)



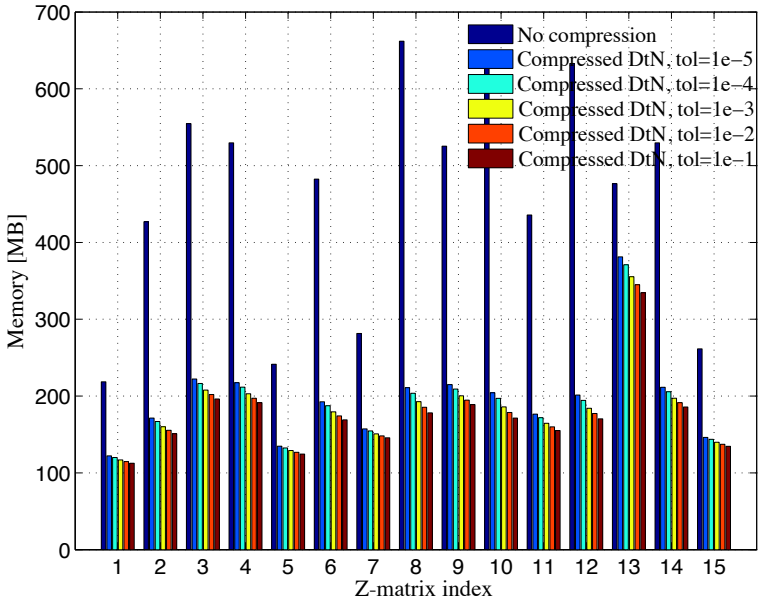
(b)

Figure 3.10. The computational geometry and its decomposition of a multi-layered printed circuit board; (a) Discretized metallic surfaces of the PCB; (b) The decomposition layout.

other bars represent the time and memory needed by the R-SVD z-matrix computation with different error tolerance. It is shown that the selective inverse/R-SVD z-matrix approach with error tolerance of 10^{-3} takes 884 seconds (in parallel) to compute and a total memory of 2.55 GB, compared to the direct method of 1445 seconds (in parallel) and 6.73 GB, this leads to a 38.8% time saving and 62% memory saving. Next the entire problem is solved with MG-FETI-LEAP [1] and IDRs [65] (L=2) iterative solver on a 10 node MacPro cluster with two 2.8 GHz Intel Xeon Quad-Core processors in each node interconnected with a gigabyte ethernet network. Fig. 3.12(a) shows the computed s-parameters relative error in dB versus simulated frequencies ($\epsilon = 10^{-3}$) by using the s-parameters that are computed with uncompressed Z-matrices as a reference. It is observed that the all the relative error of s_{11} and s_{21} fall below -90 dB. Fig. 3.12(b) plots the solution error in L_2-norm versus frequency. To further desmontrate the accuracy of the computed Z-matrices, Fig. 3.13(a) shows the convergence comparison of the proposed selective inverse/R-SVD approach with different error tolerance and that of the standard uncompressed approach used in [1]. This indicates that the approximated Z-matrices are quite accurate as the convergence does not degrade and the slopes are almost identical to the one that uses the exactly computed Z-matrices. In Fig. 3.13(b) the convergence comparison with respect to wall-time are plotted, and it is observed that one can achieve up to 21% time saving by adopting the selective inverse/R-SVD Z-matrix approach.

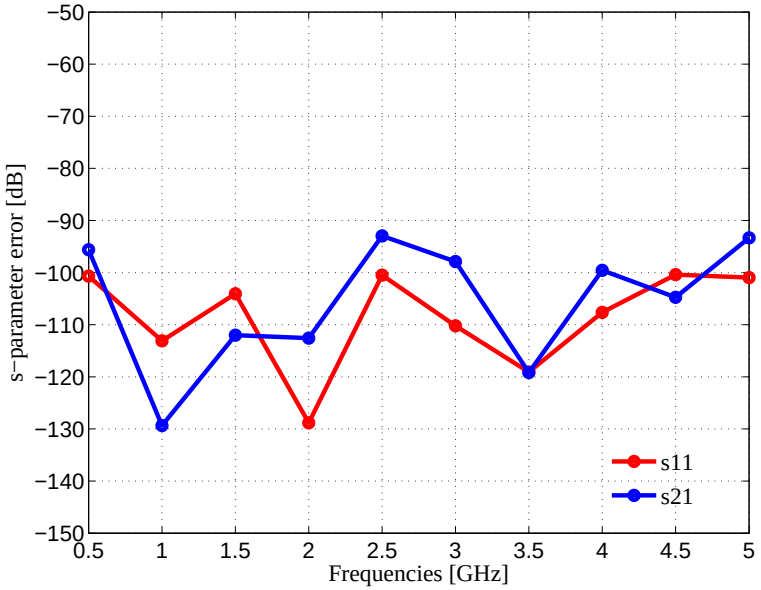


(a)

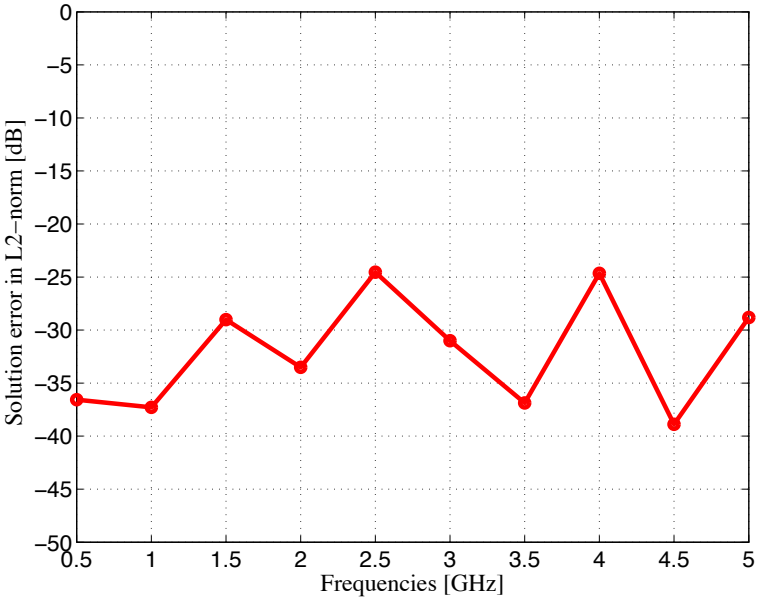


(b)

Figure 3.11. Computational savings at f=2.5 GHz from the proposed selective inverse/R-SVD assembly of the Z-matrix of each domain for various compression error tolerances, ϵ; (a) Memory comparison; (b) Time comparison.

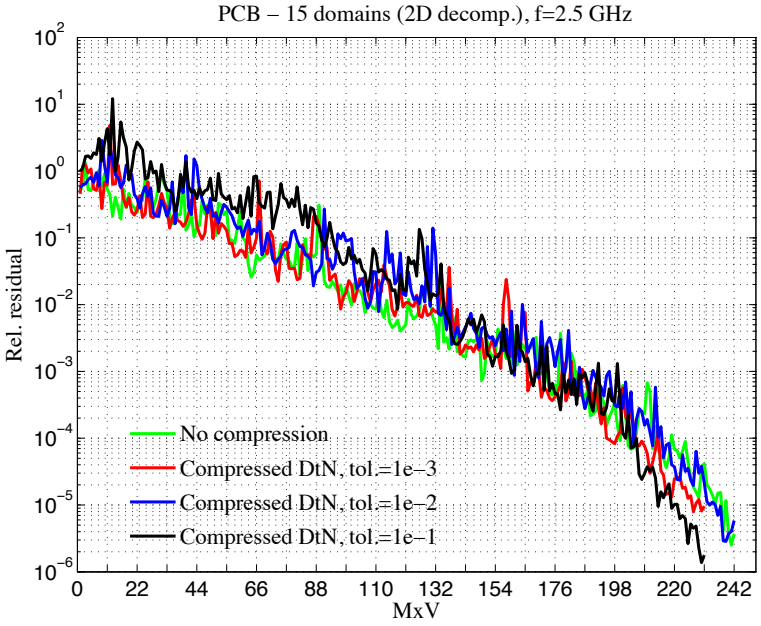


(a)

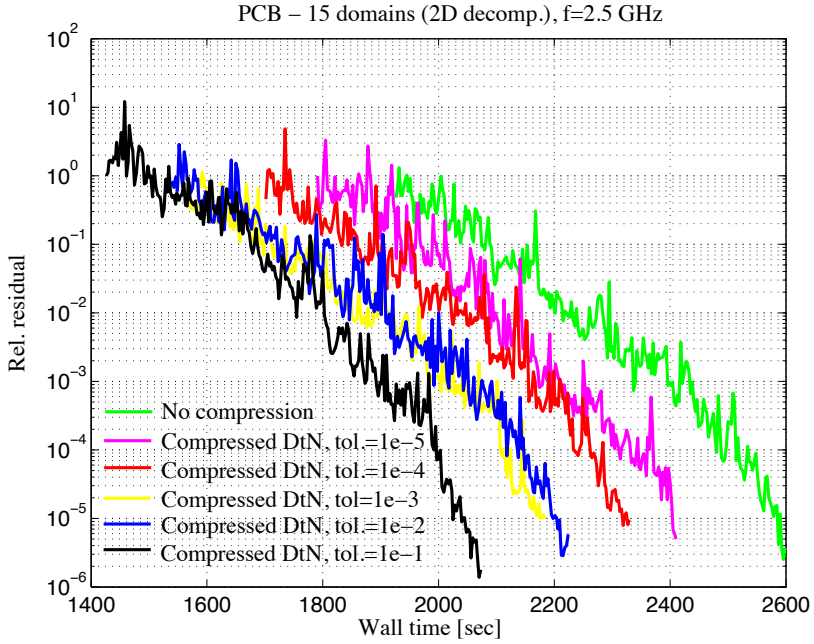


(b)

Figure 3.12. Error analysis of the PCB problem for different Z-matrix computation approach. (a) Relative s-parameters error v.s. frequency; (b) Solution error v.s. frequency.



(a)



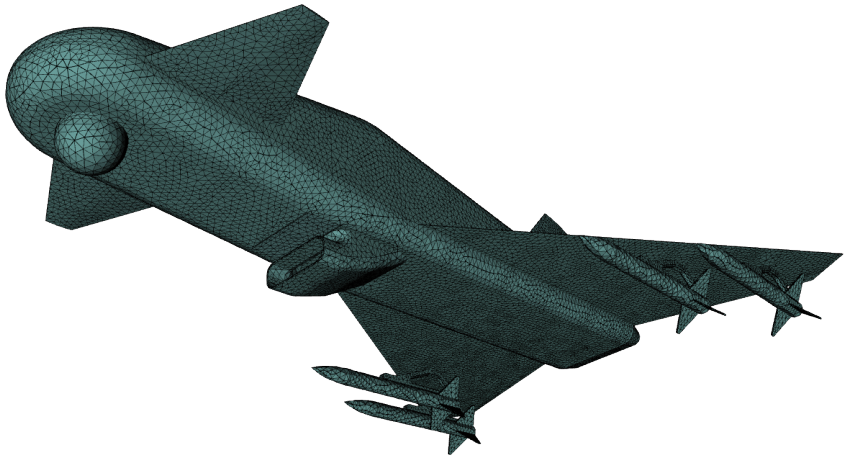
(b)

Figure 3.13. Convergence history comparison of the PCB problem for different Z-matrix computation approach. (a) Convergence history v.s. matrix-vector multiplications; (b) Convergence history v.s. time.

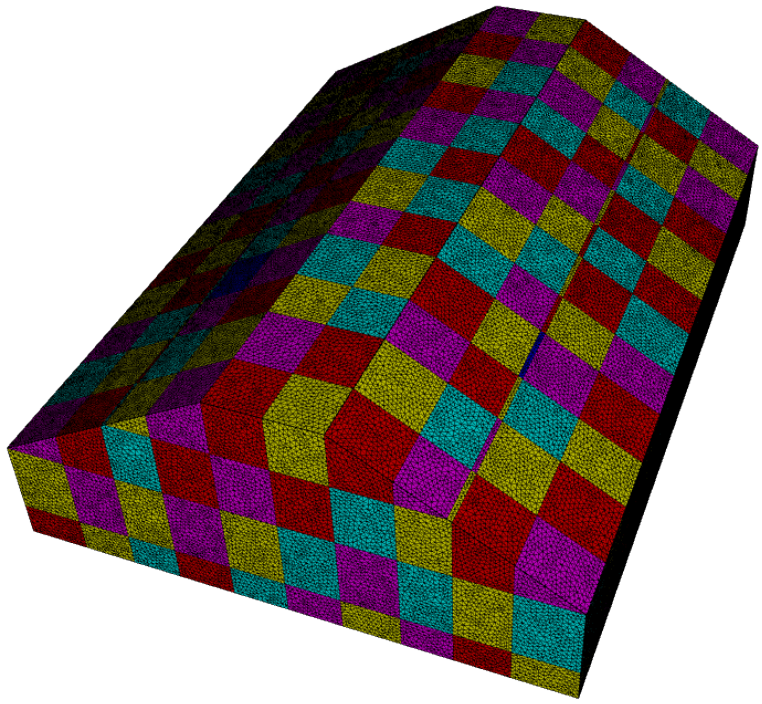
3.5.2 Scattering of A Generic Drone Aircraft

The second example is the scattering simulation of a generic drone aircraft under an oblique incident plane wave ($\theta = \phi = 45^o$) at $f = 1GHz$, which is enclosed by a 9.8m×7.3m×3.2m free-space bounding box, as shown in Fig. 3.14(a). The entire computatinal domain is discretized with 40,350,720 second-order FEM primal unknowns and 3,439,672 dual unknowns (7,624,535 tetrahedras). The problem is decomposed with 432 domains shown in Fig. 3.14(b), and solved with the Multigrid-FETI (MG-FETI) LEAP that had domain-edge and vertex LEAP buffers that span approximately $0.1\ell_{edge}$ where ℓ_{edge} is the domain interface length. The problem is solved on a in-house developed cluster with 88 CPUs with IDRs iterative solver.

We first verify the accuracy of the approximated DtN matrices. Unfortunately a direct matrix-wise error computation is computationally prohibited due to the extremely large dimension of the domain DtN matrix. In order to show the accuracy of the approximated DtN matrices, we adopt two different approaches. As what we did in the previous section, we first compare the convergence rate of two approaches that use directly computed DtN matrices and approximated ones computed by the proposed approach. Secondly, we compare the far-field pattern which is essentially the Fourier transform of the electric currents on the outer boundary of the computational domain. Fig. 3.15 shows the convergence history versus matrix-vector-product (MxV) by adopting the uncompressed and proposed approach to compute the domain DtN map operator matrices. It is shown that the convergence rates are almost identical, this indicates that the approximated domain DtN map operators are very accurate compared to the reference ones computed via the direct selective inverse approach. Fig. 3.16 shows the bistatic radar-cross-section (RCS). It is observed that the far-field pattern resulting from the approximated domain DtN map operator matrices well agrees with the reference RCS results, the relative ℓ_2 norm error is 0.251%.



(a)



(b)

Figure 3.14. A geometry view of the original generic drone model and domain decomposition; (a) a view of the geometry; (b) decomposed domains.

Fig. 3.17 shows the induced electric currents **J** on the surface of the generic drone aircraft.

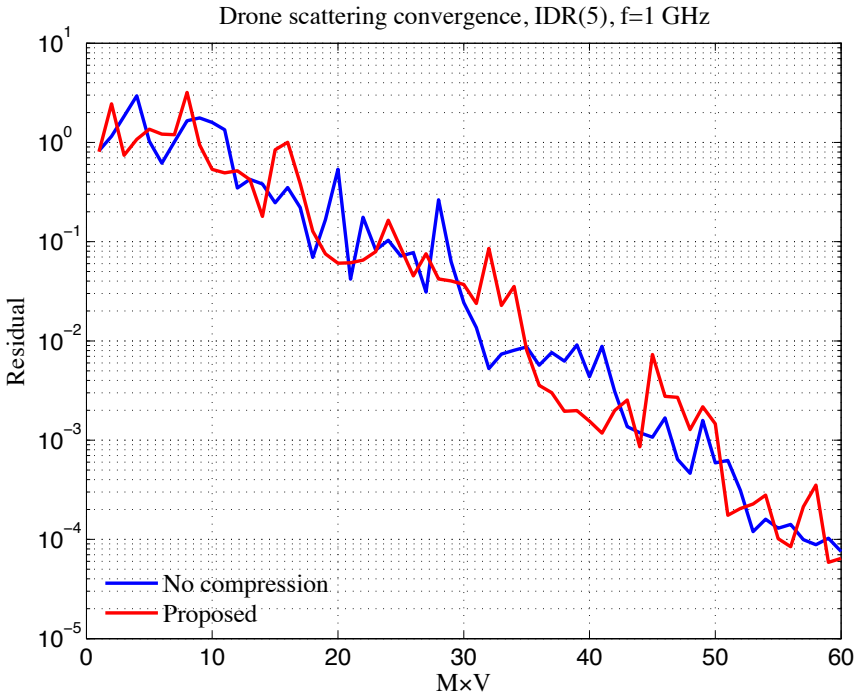


Figure 3.15. Convergence history versus matrix-vector-multiplications for the scattering simulation of a generic drone aircraft at $f = 1GHz$.

Next, we investigate the achieved efficiency of the proposed approach compared to the uncompressed approach for this challenging problem. The convergence history by adopting the direct and proposed domain DtN map operator approaches versus the run wall-time are plotted in Fig. 3.18. The proposed approach takes 3 hours 22 minutes 41 seconds to finish the simulation on a cluster with 88 CPU cores, while the direct approach takes 3 hours 52 minutes 15 seconds, a 15% total time saving is achieved. As shown earlier, the proposed approach significantly saves storage for storing the DtN matrices thanks to the nearly optimal rank finder algorithm. The total memory for DtN matrices of the proposed approach is 138.8 GB compared to 228 GB with the direct computation approach, a 30% memory saving is achieved. More details of the computational statistics are given in Table 3.2.

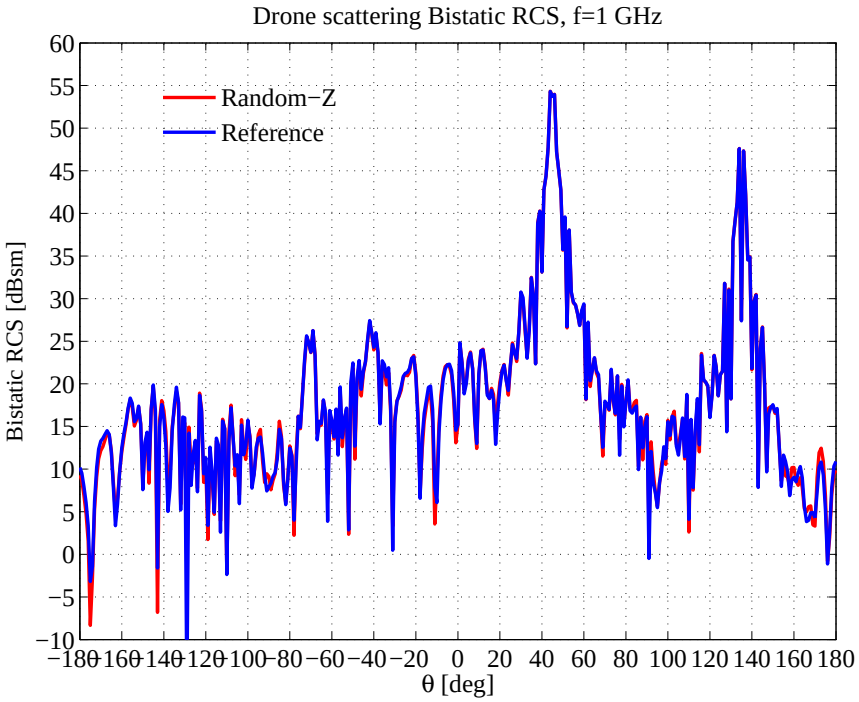


Figure 3.16. Bistatic RCS of the generic drone aircraft at $f = 1GHz$ on the incident plane ($\theta = \phi = 45^o$).

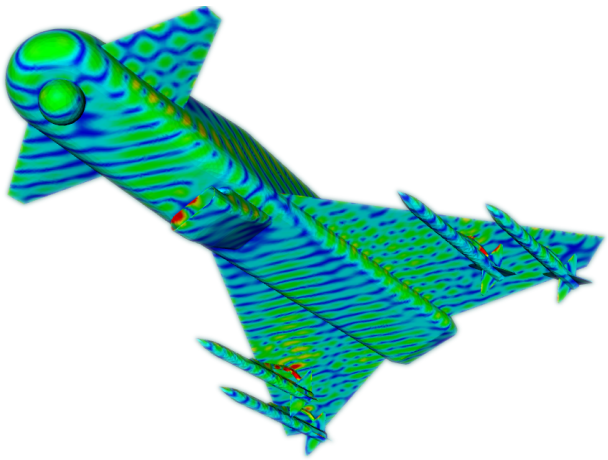


Figure 3.17. The electric currents induced on the surface of the generic drone aircraft due to an incident wave at $\phi = \theta = 45^o$ and $f = 1GHz$.

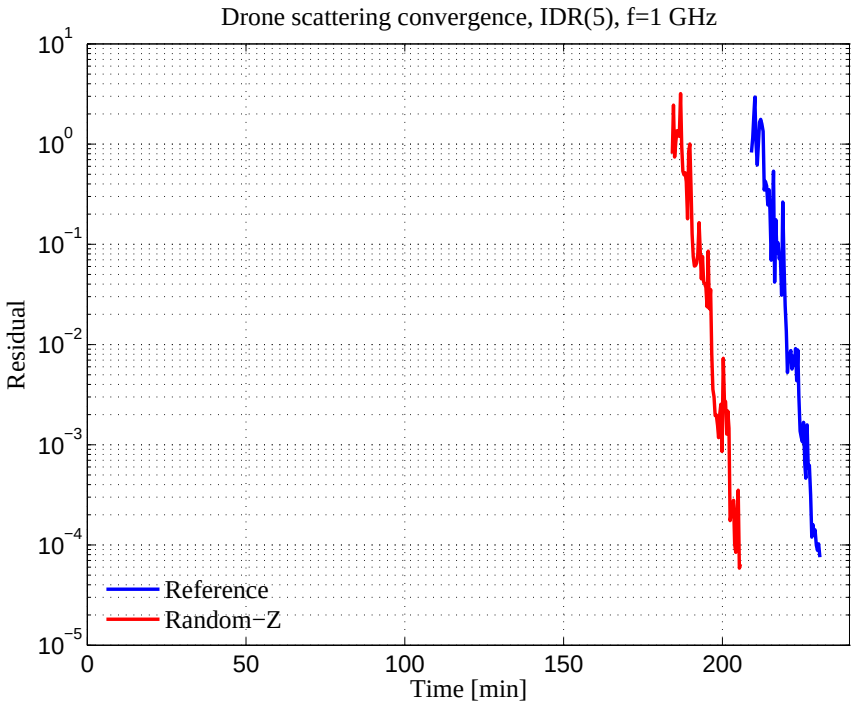


Figure 3.18. Convergence history versus run wall-time for the scattering simulation of the generic drone aircraft due to an incident wave at $\phi = \theta = 45^o$ and $f = 1GHz$.

Table 3.2. Computational statistics for the scattering simulation of a generic drone aircraft at f=1 GHz due to an oblique incident wave at $\phi = \theta = 45^o$.

Method	Unknown number (M)	MxV	Total time (h:m:s)	Z-Matrix memory [GB]	Total memory [GB]
Direct-Z	43.79	62	03:52:15	228	296.15
Random-Z		**59**	**03:22:41**	**138.8**	**208.95**

3.6 Discussion

This section introduces an computationally efficient technique to compute the domain DtN map operator in FETI-2λ DDM. The proposed method adopts an automated process to find the almost optimum rank to approximate the interaction between the EM DoFs out of two groups of surface triangles. Numerical examples suggest a 15-25% time saving and more than 30% memory saving can be achieved by adopting the proposed technique. It is important to note that the computational

time and memory cost of computing domain Z-matrices are both the dominant factors that affect the computational efficiency of the FETI-2λ FE-DDM method.

CHAPTER 4

W-FETI: A RELIABLE AND SCALABLE PRECONDITIONER FOR FETI DDM

Global information transfer is necessary for robust and scalable DDM as a DDM with only local preconditioners spreads information to neighboring domains only at each iteration. This is due to the fact that the FETI-2λ DDM coupling matrices $\mathbf{B}$ and $\mathbf{D}$ are locally defined, i.e., domain-wise. Assume a N-domain problem is considered, in order to propagate the error residual throughout the entire computational domain, the number of iteration k would be $k \sim N$. In other words, the DDM is not scalable w.r.t. the domain count.

Global preconditioning perhaps offers the most suitable computational vehicle for global communication in each iteration. The development of these types of preconditioners usually involves defining an auxilary problem that spans over the entire computational domain which is solved in each iteration. Unfortunately, for large problems with fine discretization, the subspace becomes considerably large to handle with direct solvers. Also, the efficiency and effectiveness is highly dependent on the number of global basis functions and their spatial distribution, of which both are unknown a-priori. In this chapter, we aim to develop a new global preconditioner based on the Woodbury matrix identity and FETI-2λ DDM (W-FETI). The proposed global preconditioner is expected to overcome some robustness and scalability issues encountered during applying state-of-art global preconditioners while solving challenging real-life CEM problems.

4.1 W-FETI Global Preconditioner

The W-FETI global preconditioner is inspired by the same observation that is adopted in developing the randomized computation of domain discrete DtN map operators, the sub-blocks of the discrete DtN map operator of each domain is rank deficient sourcing and receiving DoFs that are apart from each other. In W-FETI, the low rank of each domain's discrete DtN operator will be combined for the entire decomposition, and then the dominant sub-space in the resulting operator will be used to generate a global sub-space that has significantly reduced dimension compared to the original interface DoFs count. This approach algebraically and reliably incorporates the global information transfer mechanism of the underlying physical problem, thus it is expected to be robust and effective. More specifically, it is based on SVD that is "optimal" in size, can be computed efficiently from local SVDs, and automatically (through a tolerance parameter) finds the number and "shape" of global modes in each interface, resulting in a very robust and effective global preconditioner.

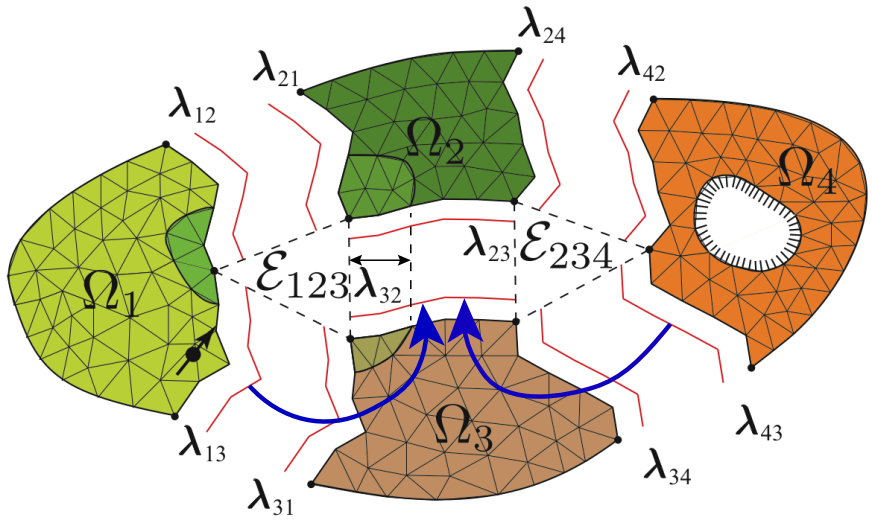


Figure 4.1. A computational model with 4 domains to illustrate the assembly of the W-FETI global preconditioner.

4.1.1 Preconditioner Assembly

To ease understanding the proposed formulation, we consider a 2D decomposition of a model problem shown in Fig. 4.1. The FETI-2λ matrix equation arising from the decomposed problem reads as

$$\left[\begin{array}{cc|cc|cc|cc|cc} \mathbf{T}_{12} & \mathbf{Q}_{12,21} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{12,23} & \mathbf{0} & \mathbf{K}_{12,24} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{Q}_{21,12} & \mathbf{T}_{21} & \mathbf{K}_{21,13} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{T}_{13} & \mathbf{Q}_{13,31} & \mathbf{0} & \mathbf{K}_{13,32} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{13,34} & \mathbf{0} \\ \mathbf{K}_{31,12} & \mathbf{0} & \mathbf{Q}_{31,13} & \mathbf{T}_{31} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{23,31} & \mathbf{T}_{23} & \mathbf{Q}_{23,32} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{23,34} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_{32,21} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{32,23} & \mathbf{T}_{32} & \mathbf{K}_{32,24} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{T}_{24} & \mathbf{Q}_{24,42} & \mathbf{0} & \mathbf{K}_{24,43} \\ \mathbf{0} & \mathbf{K}_{42,21} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{42,23} & \mathbf{0} & \mathbf{K}_{42,24} & \mathbf{T}_{42} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{34,42} & \mathbf{T}_{34} & \mathbf{Q}_{34,43} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{43,31} & \mathbf{0} & \mathbf{K}_{43,32} & \mathbf{0} & \mathbf{0} & \mathbf{Q}_{43,34} & \mathbf{T}_{43} \end{array}\right] \left[\begin{array}{c} \boldsymbol{\lambda}_{12} \\ \boldsymbol{\lambda}_{21} \\ \hline \boldsymbol{\lambda}_{13} \\ \boldsymbol{\lambda}_{31} \\ \hline \boldsymbol{\lambda}_{23} \\ \boldsymbol{\lambda}_{32} \\ \hline \boldsymbol{\lambda}_{24} \\ \boldsymbol{\lambda}_{42} \\ \hline \boldsymbol{\lambda}_{34} \\ \boldsymbol{\lambda}_{43} \end{array}\right] = \left[\begin{array}{c} \mathbf{g}_{12} \\ \mathbf{g}_{21} \\ \hline \mathbf{g}_{13} \\ \mathbf{g}_{31} \\ \hline \mathbf{g}_{23} \\ \mathbf{g}_{32} \\ \hline \mathbf{g}_{24} \\ \mathbf{g}_{42} \\ \hline \mathbf{g}_{34} \\ \mathbf{g}_{43} \end{array}\right]. \quad (4.1)$$

To devise an effective and efficient algebraic global preconditioning strategy for (4.1), one must carefully study the reduced matrix equation (Schur-complement of LMs) that can be simply denoted as,

$$\mathbf{F}\,\lambda = \mathbf{g}, \quad (4.2)$$

where $\mathbf{F} = \mathbf{T} - \sum_{i=1}^{N} \mathbf{B}_i \mathbf{A}_i^{-1} \mathbf{D}_i \in \mathbb{C}^{n \times n}$. Although this form is simple, it provides little or no preconditioning intuition. Alternatively,

$$\mathbf{F} = \mathbf{D} + \mathbf{K} \quad (4.3)$$

offers a better choice. $\mathbf{D} = blockDiag(\mathbf{F})$ is the block diagonal part of the interface-based partitioned $\mathbf{F}$, and $\mathbf{K} = \mathbf{F} - \mathbf{D}$ is block-wise sparse with zero diagonal blocks. This decomposition is motivated by the fact that $\mathbf{K}$ has zero diagonal blocks and its off-diagonal blocks directly involve the discrete DtN map operator of the domains,

but for different (non-self) interfaces. As shown in the previous chapter, each $\mathbf{K}_{ij,mn}$ in (4.1) is rank deficient as $\mathbf{K}_{ij,mn}$ represents the interactions of LMs on different interface $\mathcal{I}_{ij}$ and $\mathcal{I}_{mn}$, where $\{ij\} \neq \{mn\}$. Therefore, it can be concluded that $\mathbf{K}$ is also rank deficient, meaning it can be approximated by an efficient matrix decomposition method, e.g., low-rank SVD. By leveraging this low rank observation, one can rewrite the inverse of $\mathbf{F}$ as

$$\mathbf{F}^{-1} = (\mathbf{D} + \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^*)^{-1}. \tag{4.4}$$

At this pooint (4.4) is exact but numerically intractable, as it does not take advantage of the low-rank properties of DtN map operators. To do that, the skinny SVD of $\mathbf{K}$

$$\mathbf{K} = \tilde{\mathbf{U}}\,\tilde{\mathbf{\Sigma}}\,\tilde{\mathbf{V}}^* \equiv \mathrm{svd}_k(\mathbf{K})(k \ll n) \tag{4.5}$$

is used to generate a low-rank approximation of $\mathbf{F}^{-1}$, that in turn can be used as a preconditioner,

$$\mathbf{M}_{\mathcal{W}}^{-1} = (\mathbf{D} + \tilde{\mathbf{U}}\,\tilde{\mathbf{\Sigma}}\,\tilde{\mathbf{V}}^*)^{-1}. \tag{4.6}$$

The inverse computation in this preconditioner is best performed by invoking the Woodbury matrix identity [73],

$$\mathbf{M}_{\mathcal{W}}^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\,\tilde{\mathbf{U}}(\tilde{\mathbf{\Sigma}}^{-1} + \tilde{\mathbf{V}}^*\,\mathbf{D}^{-1}\,\tilde{\mathbf{U}})^{-1}\,\tilde{\mathbf{V}}^*\,\mathbf{D}^{-1}, \tag{4.7}$$

where the seemingly time and memory consuming task of inverting $(\tilde{\mathbf{\Sigma}}^{-1} + \tilde{\mathbf{V}}^*\,\mathbf{D}^{-1}\,\tilde{\mathbf{U}}) \in \mathbb{C}^{k\times k}$ can be done efficiently due to the orders of magnitude smaller k than n, and the sparse nature of $\mathbf{K}$.

Equation (4.7) is the core formula of the proposed global preconditioner. It comprises a global preconditioning component in the form of $(\tilde{\mathbf{\Sigma}}^{-1} + \tilde{\mathbf{V}}^*\,\mathbf{D}^{-1}\,\tilde{\mathbf{U}})^{-1}$, and many local ones $\mathbf{D}_i^{-1}, i = 1, 2, \cdots$, that happens to be identical to the locally exact algebraic preconditioner (LEAP[1]) proposed in [1].

The numerical computation of the skinny SVD, $\text{svd}_k(\mathbf{K})$, remains prohibitively large even for moderate size DD problems or using state-of-art rank-revealing SVD algorithm. Indeed, the challenge of the W-FETI preconditioner is how to efficiently compute such SVD for multi-million unknown problems. The key to efficient computation of $\text{svd}_k(\mathbf{K})$ arises from two observations, the sparse decomposition (or assembly) of $\mathbf{K}$ and the randomized rank-revealing SVD algorithm that is introduced in the previous chapter. We discuss these two perspectively as below.

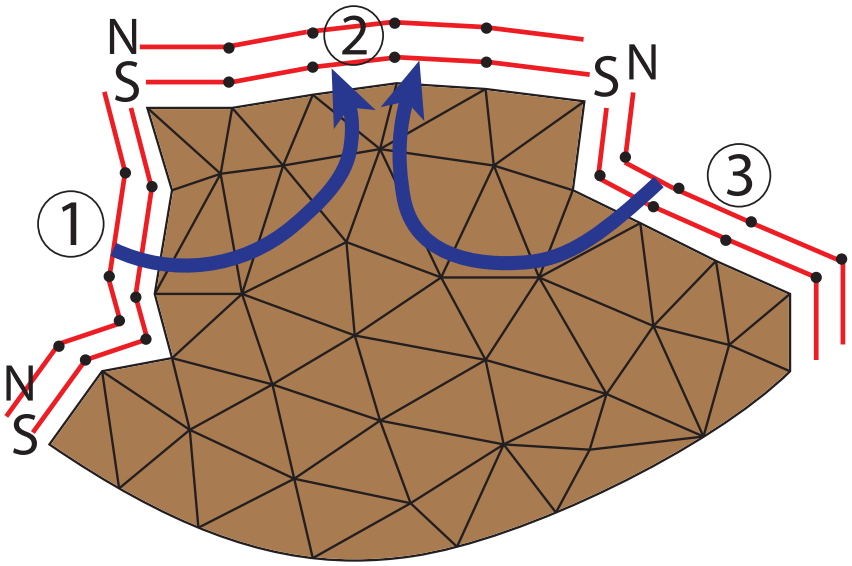


Figure 4.2. Local LM numbering scheme for domain Ω_i. (N: neighbor interface, S: self interface)

4.1.1.1 Sparse Assembly of K

Based on the sparsity property of $\mathbf{K}$, we can decompose $\mathbf{K}$ onto "domain" matrices, in a similar fashion to the FEM assembly procedure from element-matrices, namely,

$$\mathbf{K} = \sum_{i=1}^{N} \mathbf{G}^{(i)T} \mathbf{K}_D^{(i)} \mathbf{G}^{(i)}, \tag{4.8}$$

where $\mathbf{K}_D^{(i)} \in \mathbb{C}^{(n_S^{(i)}+n_N^{(i)})\times(n_S^{(i)}+n_N^{(i)})}$ is the "element"-matrix of domain i, with $n_S^{(i)}$, $n_N^{(i)}$ being the total number of "self" and "neighbor" set of redundant LM on the interfaces

on domain $i$and $\mathbf{G}^{(i)} \in \mathbb{N}^{(n_S^{(i)}+n_N^{(i)})\times n}$ represents the binary domain local-to-global mapping of LMs. To make this process more clear, we consider an individual domain Ω_i with three interfaces shown in Fig. 4.2. Note that the concerned domain is identical to Ω_3 in the decomposition problem shown in Fig. 4.1. The circle numbering is the local interface index with 'N' denoting the neighboring one and 'S' denoting the self one. The domain matrix $\mathbf{K}_D$ associated with this domain can be written as

$$\mathbf{K}_D^{(i)} = \left[\begin{array}{cc|cc|cc} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{12}^{(3)} & \mathbf{0} & \mathbf{K}_{13}^{(3)} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{21}^{(3)} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_{23}^{(3)} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{31}^{(3)} & \mathbf{0} & \mathbf{K}_{32}^{(3)} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\right], \tag{4.9}$$

where $\mathbf{K}_{mn}^{(i)}$ represents interactions between interface $\mathcal{I}_m$ and $\mathcal{I}_n$ and is low-rank. The mapping matrix $\mathbf{G}^{(i)}$ is block-wise binary and very sparse mapping local redundant LMs to global ones. For domain i, it relates

$$\left[\begin{array}{c} \boldsymbol{\lambda}_{1L}\ \boldsymbol{\lambda}_{1R} \mid \boldsymbol{\lambda}_{2L}\ \boldsymbol{\lambda}_{2R} \mid \cdots \mid \boldsymbol{\lambda}_{nL}\ \boldsymbol{\lambda}_{nR} \end{array}\right]^{\mathrm{T}} = \mathbf{G}^{(i)} . \left[\begin{array}{c} \boldsymbol{\lambda}_{1_S}^{(i)}\ \boldsymbol{\lambda}_{1_N}^{(i)} \mid \boldsymbol{\lambda}_{2_S}^{(i)}\ \boldsymbol{\lambda}_{2_N}^{(i)} \mid \boldsymbol{\lambda}_{3_S}^{(i)}\ \boldsymbol{\lambda}_{3_N}^{(i)} \mid \boldsymbol{\lambda}_{4_S}^{(i)}\ \boldsymbol{\lambda}_{4_N}^{(i)} \end{array}\right]^{\mathrm{T}}, \tag{4.10}$$

where $\boldsymbol{\lambda}_{mL}$ and $\boldsymbol{\lambda}_{mR}$ are the left and right LM sets of interface $\mathcal{I}_m$. The resulting mapping of the local $\mathbf{K}_{mn}^{(3)}$ to global $\mathbf{K}_{ij,uv}$ matrix defined in (4.1) is given in Table 4.1.

Table 4.1. A example mapping of element matrix $\mathbf{K}$ to globally defined $\mathbf{K}$ in FETI-2λ for the domain shown in Fig. 4.2.

Local $\mathbf{K}$	$\mathbf{K}_{12}^{(3)}$	$\mathbf{K}_{13}^{(3)}$	$\mathbf{K}_{21}^{(3)}$	$\mathbf{K}_{23}^{(3)}$	$\mathbf{K}_{31}^{(3)}$	$\mathbf{K}_{32}^{(3)}$
Global $\mathbf{K}$	$\mathbf{K}_{13,32}$	$\mathbf{K}_{13,34}$	$\mathbf{K}_{23,31}$	$\mathbf{K}_{23,34}$	$\mathbf{K}_{43,31}$	$\mathbf{K}_{43,32}$

Therefore, one can show using the binary nature and column linear independence of $\mathbf{G}^{(i)}$ and the sparse structure of $\mathbf{K}^{(i)}$ that

$$\mathrm{svd}_k(\mathbf{K}) = \sum_{i=1}^{N} \mathbf{G}^{(i)T} \mathrm{svd}_{k_i}(\mathbf{K}_D^{(i)})\, \mathbf{G}^{(i)} \tag{4.11}$$

where $\sum_{i=1}^{N} k_i = k$. This property provides an elegant and efficient way of obtaining $\tilde{\mathbf{U}}$, $\tilde{\mathbf{\Sigma}}$ and $\tilde{\mathbf{V}}$ in (4.7) by performing many manageable-size SVDs.

4.1.1.2 Randomized SVD

Although $\mathbf{K}_D^{(i)}$ is significantly smaller compared to $\mathbf{K}$ in size, a direct application of $\mathrm{svd}(\mathbf{K}_D^{(i)})$ with complexity of $\mathcal{O}(n^3)$ is still computationally prohibitive. This is because the dimension of $\mathbf{K}_D^{(i)}$ equals to the size of LMs for domain i, in a realistic large-scale simulation, the domain LMs count can easily reach 100k. Alternatively, by leveraging the rank deficiency of $\mathbf{K}_D^{(i)}$ and following the same spirit arising from the algorithms developed in the previous chapter, $\mathbf{K}_D^{(i)}$ can be sought as

$$\mathrm{svd}_{k_i}(\mathbf{K}_D^{(i)}) = \mathrm{svd}_{k_i}(\mathbf{L}\,\mathbf{R}) = \mathbf{L}\,\mathrm{svd}_{k_i}(\mathbf{R}) = \tilde{\mathbf{U}}_{k_i}\,\tilde{\mathbf{\Sigma}}_{k_i}\,\tilde{\mathbf{V}}_{k_i}^{*}, \tag{4.12}$$

where $\mathbf{L} \in \mathbb{C}^{n_D \times k_i}$ and $\mathbf{R} \in \mathbb{C}^{k_i \times n_D}$ are low-rank approximation factors of $\mathbf{K}_D^{(i)}$, n_D is size of LMs for domain Ω_i, $k_i \ll n_D$. Low-rank factors $\mathbf{L}$ and $\mathbf{R}$ can be solved by the proposed randomized SVD algorithm in Algorithm 4.

The proposed SVD approach in Algorithm 4 relates

$$\mathbf{L} = \mathbf{Q}, \tag{4.13}$$

$$\mathbf{R} = \mathbf{B}. \tag{4.14}$$

This enables to apply a direct SVD application of a super skinny matrix $\mathbf{B} \in \mathbb{C}^{k_i \times n_D}$, $k_i \ll n_D$. The size of k_i is increased iteratively by k at each adaptive pass. Thus, the

Algorithm 4 : Adaptive randomized SVD for a rank deficient matrix

INPUT:
$\mathbf{K}_D^{(i)}$: "Element" matrix of FETI-2λ reduced system.
p: Incremental column size of random matrix
ε: Target error residual to stop the adaptive process
ε_{svd}: Global singular values truncation tolerance

OUTPUT:
$\tilde{\mathbf{U}}_{k_i}, \tilde{\mathbf{\Sigma}}_{k_i}, \tilde{\mathbf{V}}_{k_i}^*$: Low-rank SVD decomposition of $\mathbf{K}_D^{(i)}$.

$q = 0$
while $\max\{\|\mathbf{e}_i\|, i = 1, 2, \cdots\} > \varepsilon$ **do**
  $q \leftarrow q + 1$
  `Generate a` $n_D \times p$ `random matrix` $\mathbf{\Omega}$.
  `Form` $\mathbf{A} = \mathbf{K}_D^{(i)} \mathbf{\Omega}$.
  `Compute QR factorization` $\mathbf{A} = \mathbf{Q}_q \mathbf{R}$.
  $\mathbf{Q} \leftarrow [\mathbf{Q}\ \mathbf{Q}_q]$.
  `Construct error estimator` $\mathbf{e} = (\mathbf{I} - \mathbf{Q}\mathbf{Q}^*)\,\mathbf{A}$.
end while
`Form` $\mathbf{B} = \mathbf{Q}^* \mathbf{K}_D^{(i)}$.
`Compute SVD` $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$.
`Truncate singular values` $\forall \sigma \in \mathbf{\Sigma}, \sigma/\sigma_{\max} > \varepsilon_{\text{svd}}, \mathbf{B} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$.
`Set` $\tilde{\mathbf{U}}_{k_i} = \mathbf{Q}\tilde{\mathbf{U}}, \tilde{\mathbf{\Sigma}}_{k_i} = \tilde{\mathbf{\Sigma}}, \tilde{\mathbf{V}}_{k_i}^* = \tilde{\mathbf{V}}^*$.

final size of the constructed W-FETI global preconditioner is

$$k = \sum_{i=1}^{N} count(\forall \sigma \in \tilde{\mathbf{\Sigma}}_{k_i}, \sigma/\sigma_{\max} > \epsilon_{\text{svd}}) \tag{4.15}$$

The proposed randomized SVD approach combined with the sparse assembly of $\mathbf{K}$ offers an intelligent way to automatically pick the number and shape of global modes used to represent each interface in the global preconditioning subspace. The final size of the global preconditioner can be controlled by two important parameters, the adaptive randomized SVD algorithm error tolerance ε and the singular values truncation tolerance ε_{svd}. This is a significant advantage over multigrid or plane wave based global preconditioning methods, that require a priori knowledge of the number of preconditioning modes and use ad-hoc mode sets in each interface. Selected numerical examples in the upcoming sections will be used to illustrate this numerical strength.

Remark

1. The choice of the maximum singular value $\sigma_{\max}$ to normalize all the singular values is critical since it considerably affects the quality of the constructed subspace of the global modes. One option is to use the maximum singular value $\sigma^i_{\max}$ arising from each SVD to perform truncation on a domain/element basis, whereas the other option is to choose

$$\sigma_{\max} = \max\{\sigma^i_{\max}, i = 1, 2, \cdots, N\} \tag{4.16}$$

in a global perspective. Numerical results show that the choice of (4.16) brings better preconditioned system. This can be explained that a sub-block matrix of $\mathbf{K}$ in the FETI-2λ represents the DoFs interactions of two interfaces coinciding at a domain edge, therefore some of sub-blocks represent a strong coupling whereas others are weak. If we use each domain's $\sigma_{\max}$ as the normalization creteria, dominant modes may accidentally be dumped for those sub-blocks representing strong coupling whereas unnecessary modes maybe kept for thoses of weak coupling. By using a global chosen $\sigma_{\max}$, it assures to keep the most dominant modes contributed to the global subspace.

2. In (4.7), the inverse of the global matrix $(\tilde{\mathbf{\Sigma}}^{-1} + \tilde{\mathbf{V}}^* \mathbf{D}^{-1} \tilde{\mathbf{U}})^{-1}$ and local ones $\mathbf{D}^{-1}$ are never explicitly formed since in an interative solver only the matrix vector multiplication is needed.

3. The assembly and following randomized SVD of each "element" matrix $\mathbf{K}^{(i)}_D$ is an embarrassingly parallel process because it only needs the locally defined matrices $\mathbf{B}, \mathbf{Z}, \mathbf{D}$ which happen to be associated with the same domain.

4. The parallel scalability of assembling the W-FETI global preconditioner is only hampered while constructing the global matrix $(\tilde{\mathbf{\Sigma}}^{-1} + \tilde{\mathbf{V}}^* \mathbf{D}^{-1} \tilde{\mathbf{U}})$ as it involves matrix-matrix products from different domains and interfaces. Achieved parallel scalability of W-FETI will be illustrated in the numerical study section and next chapter.

4.1.2 Integration with LEAP

The combination of local and global preconditioners would provide better matrix preconditioning than applying local or global preconditioner alone. In this work, the W-FETI preconditioner is integrated with the second and third level local exact algebraic precondtioners (LEAP) proposed in [1]. The integration of W-FETI with LEAP^2 and LEAP^3 is implemented in a multiplicative fashion [74]. The complete W-FETI preconditioning procedure at all levels including the local preconditioning is given as below,

$$\begin{aligned} \mathbf{u} &\longleftarrow \mathbf{M}_{\mathcal{W}}^{-1}\mathbf{r}, \\ \mathbf{u} &\longleftarrow \mathbf{u} + \mathbf{M}_{\mathcal{E}}^{-1}(\mathbf{r} - \mathbf{F}\mathbf{u}), \\ \mathbf{u} &\longleftarrow \mathbf{u} + \mathbf{M}_{\mathcal{V}}^{-1}(\mathbf{r} - \mathbf{F}\mathbf{u}), \end{aligned} \tag{4.17}$$

where $\mathbf{u}$ is the search direction vector at each iteration of the iterative solver and $\mathbf{r}$ is the residual vector. $\mathbf{M}_{\mathcal{W}}^{-1}$ is given in (4.7) and $\mathbf{M}_i^{-1}(i = 1, 2, 3)$ represents the ith level preconditioner matrix, $\mathbf{M}_{\mathcal{E}}^{-1}$ and $\mathbf{M}_{\mathcal{V}}^{-1}$ are the LEAP^2 (domain-edge) and LEAP^3 (domain-vertex) local precondtioner, respectively. Details about assembling these local preconditioners can be found in [1]. The three level preconditioners in matrix form are,

$$\begin{aligned} &\text{Level 1: } \mathbf{M}_1^{-1} = \mathbf{M}_{\mathcal{W}}^{-1}, \\ &\text{Level 2: } \mathbf{M}_2^{-1} = \mathbf{M}_{\mathcal{E}}^{-1} + \mathbf{M}_1^{-1} - \mathbf{M}_{\mathcal{E}}^{-1}\,\mathbf{F}\,\mathbf{M}_1^{-1}, \\ &\text{Level 3: } \mathbf{M}_3^{-1} = \mathbf{M}_{\mathcal{V}}^{-1} + \mathbf{M}_2^{-1} - \mathbf{M}_{\mathcal{V}}^{-1}\,\mathbf{F}\,\mathbf{M}_2^{-1}. \end{aligned} \tag{4.18}$$

Remark

1. The goal of LEAP^2 local preconditioner is to precondition the strong coupling of corner Lagrange multipliers around domain edges, it define local problems around domain edges that helps to capture much of the underlying physics by introducing a

cylinder centered around each domain edge with a user specified radius (LEAP^2 buffer region). It is reported in [1] that in order to achieve reasonable convergence performance, the MG-LEAP preconditioning scheme often requires to set large LEAP^2 buffers at each domain edge, which significantly increases the computational effort both in time and memory to compute the LEAP^2 preconditioner matrices. More importantly the buffer size of LEAP^2 in [1] is not known a-priori and could be different in different domain edges, thus it renders the approach cumbersome. In the proposed W-FETI global preconditioner, the preconditioning of strong coupling of LMs around domain edges are automatically taken care of by picking the dominant singular values and its corresponding singular vectors of the DtN, therefore the proposed preconditioning could use a much smaller LEAP^2 buffer region while achieve same or better convergence performance as MG-FETI-LEAP. This important reusability and robustness advantage will be demonstrated with numerical results in the following sections and next chapter.

2. The proposed W-FETI-LEAP preconditioning scheme has only up to three preconditioning levels thanks to the fact that the LEAP^1 preconditioner is already integrated into the W-FETI global precondtioner as shown in (4.7), while the multigrid (MG)-LEAP approach proposed in [1] has up to four levels.

3. The global FETI matrix $\mathbf{F}$ is not involved in the matrix-vector multiplication stage if only the first level local preconditioner LEAP^1 is needed. This is a significant advantage since a matrix product with the global FETI matrix $\mathbf{F}$ involves tremendous amount of data communication in a parallel computing environment. However, this is not the case for MG-FETI-LEAP^1.

4.2 Numerical Study

In this section, we present multiple numerical examples to showcase the effectiveness and robustness of the proposed W-FETI global preconditioner. In all examples

that involve local preconditioners, the minimum LEAP^p buffer $< 10\%$ of each interface is used.

4.2.1 Eigenspectra of W-FETI with LEAP

In this section, we consider a general parallel waveguide model that has PEC boundaries on the top and bottom, and PMC on the left and right sides with first-order absorbing boundary condition on front and back. We will show the eigenspectra of the preconditioned FETI-2λ global matrix $\mathbf{F}$ for 1D, 2D and 3D decomposition cases.

We first consider a one-way decomposition problem. The length of the waveguide is 5λ and cross section is measured as $1\lambda \times 1\lambda$ at 50 MHz. The model is discretized with 67,000 second-order FE unknowns and decomposed into 5 domains positioned in one-way, where each domain is a 1m$\times$1m$\times$1m free-space cube, shown in Fig. 4.3(a). Fig. 4.3(b) shows the eigenspectrum of the non-preconditioned FETI-2λ matrix $\mathbf{F}$, Fig. 4.3(c) shows the eigenspectrum of preconditioned system with MG-FETI-LEAP1 preconditioners whereas Fig. 4.3(d)-(e) show the eigenspectrum with W-FETI-LEAP1 under two different SVD truncation tolerance, namely, $\varepsilon_{\text{svd}} = 10^{-1}$ and $\varepsilon_{\text{svd}} = 10^{-2}$. It is observed that the multiplicative combination of local and global preconditioners provide a very small condition number and clustered eigenspectrum that is around the unity for both MG-FETI and W-FETI case. It is also expected that an even smaller condition number with more clustered eigenspectrum can be achived with smaller truncation tolerance, as shown in Fig. 4.3(e).

Now the same problem with a 2D decomposition is considered. topology. The same discretization for each domain as in 1D case is applied. The model is decomposed into 16 domains with a 4×4 decomposition topology as shown in Fig. 4.4(a). Fig. 4.4(b) shows the eigenspectrum of FETI-2λ matrix with no any preconditioners applied, Fig. 4.4(c) shows the eigenspectrum with MG-FETI-LEAP2 whereas

Fig. 4.4(d-e) show the eigenspectrum for W-FETI with different SVD truncation tolerance. Both approaches achieve a fairly small condition number and clustered eigenspectrum around unity. It is also expected that a smaller condition number and a more clustered eigenspectrum can be achieved by using a smaller singular value truncation tolerance($\varepsilon_{\text{svd}} = 10^{-2}$), as shown in Fig. 4.4(e).

Finally we consider the same problem with a 3D decomposition. The same discretization for each domain as in 1D and 2D cases is applied. The model is decomposed into 27 domains with a $3 \times 3 \times 3$ decomposition topology as shown in Fig. 4.5(a). Fig. 4.5(b) shows the eigenspectrum of the original FETI-2λ matrix, Fig. 4.5(c) shows the eigenspectrum of $\mathbf{F}$ with MG-FETI-LEAP2 whereas Fig. 4.5(d-e) show the eigenspectrum for W-FETI-LEAP2 with two different SVD truncation tolerance. It is observed that both methods still achieve considerably smaller condition number for the preconditioned system and clustered eigenspectrum. Smaller SVD tolerance leads to a more clustered eigenspectrum thus smaller condition number. It is also observed that the condition number of the preconditioned system deteriorates as the decomposition topology changes from one-way to three-dimensional case. This is expected as in a 3D decomposition, the DoFs interaction at domain edges and domain vertex becomes more strong making the system more difficult to precondition.

In order to showcase the robustness of the proposed preconditioner, we consider a simgple microstrip crossbar structure as shown in Fig. 4.6(a). The model is decomposed in a 2D decomposition topology with 9 domains. With thin PEC layer across the domain, the fields at the domain interface become more singular, and are difficult to approximate by using the ad-hoc basis functions such as used in MG-FETI formulation. Fig. 4.6(b)-(c) show the comparison of eigenspectrum of the FETI-2λ matrix after applying MG-FETI-LEAP2 and W-FETI-LEAP2 preconditioners, respectively. It is clear to see the superity of the proposed approach over MG-FETI that the eigenvalues are more clustered around the unity, this brings down the condition num-

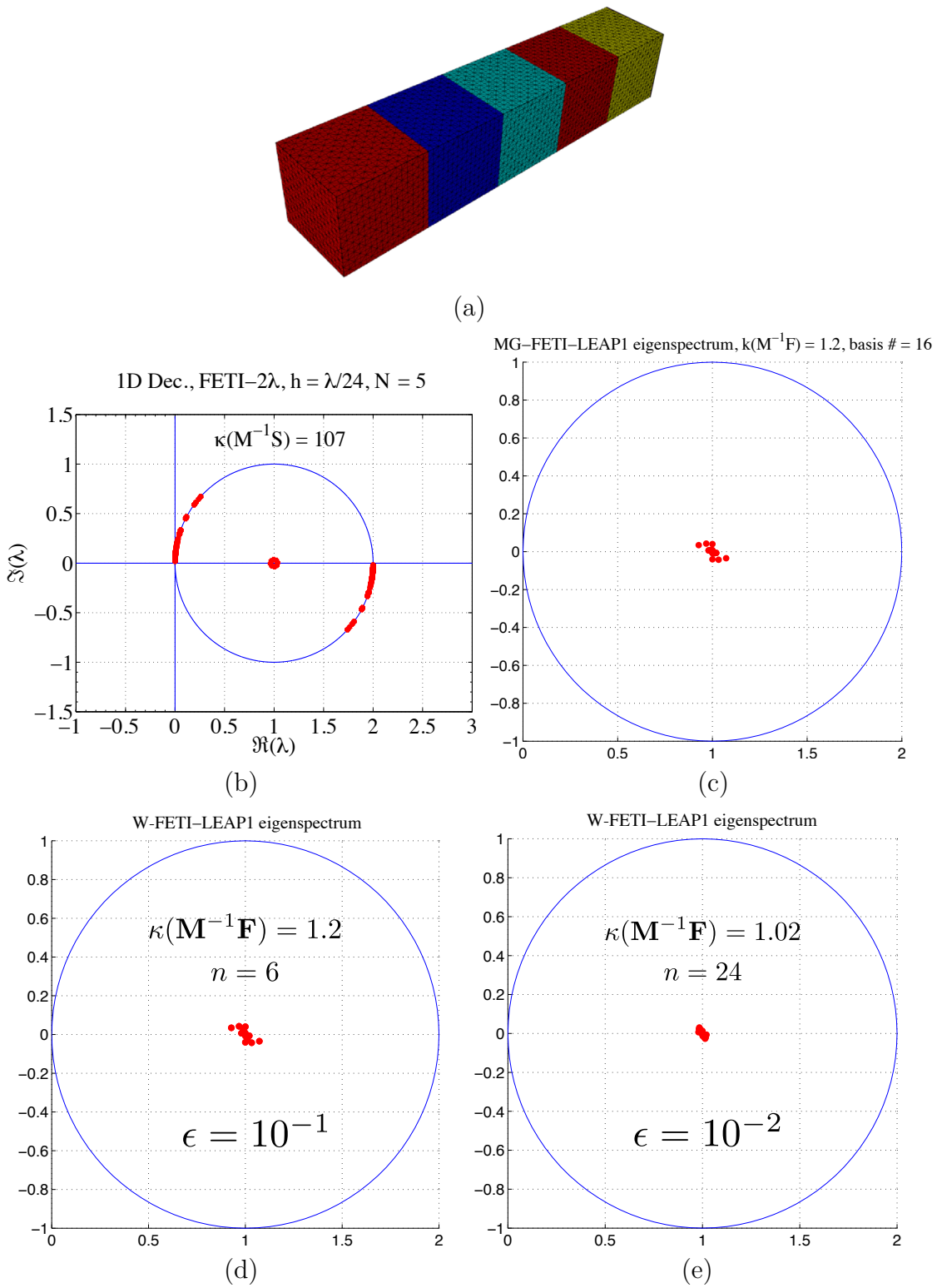


Figure 4.3. A 1D parallel waveguide example with 5 domains: (a) Geometry; (b) Eigenspectrum of the original FETI-2λ matrix; (c) Eigenspectrum with MG-FETI-LEAP1; (d) Eigenspectrum with W-FETI-LEAP1, $\varepsilon_{\text{svd}} = 10^{-1}$; (e) Eigenspectrum with W-FETI-LEAP1, $\varepsilon_{\text{svd}} = 10^{-2}$.

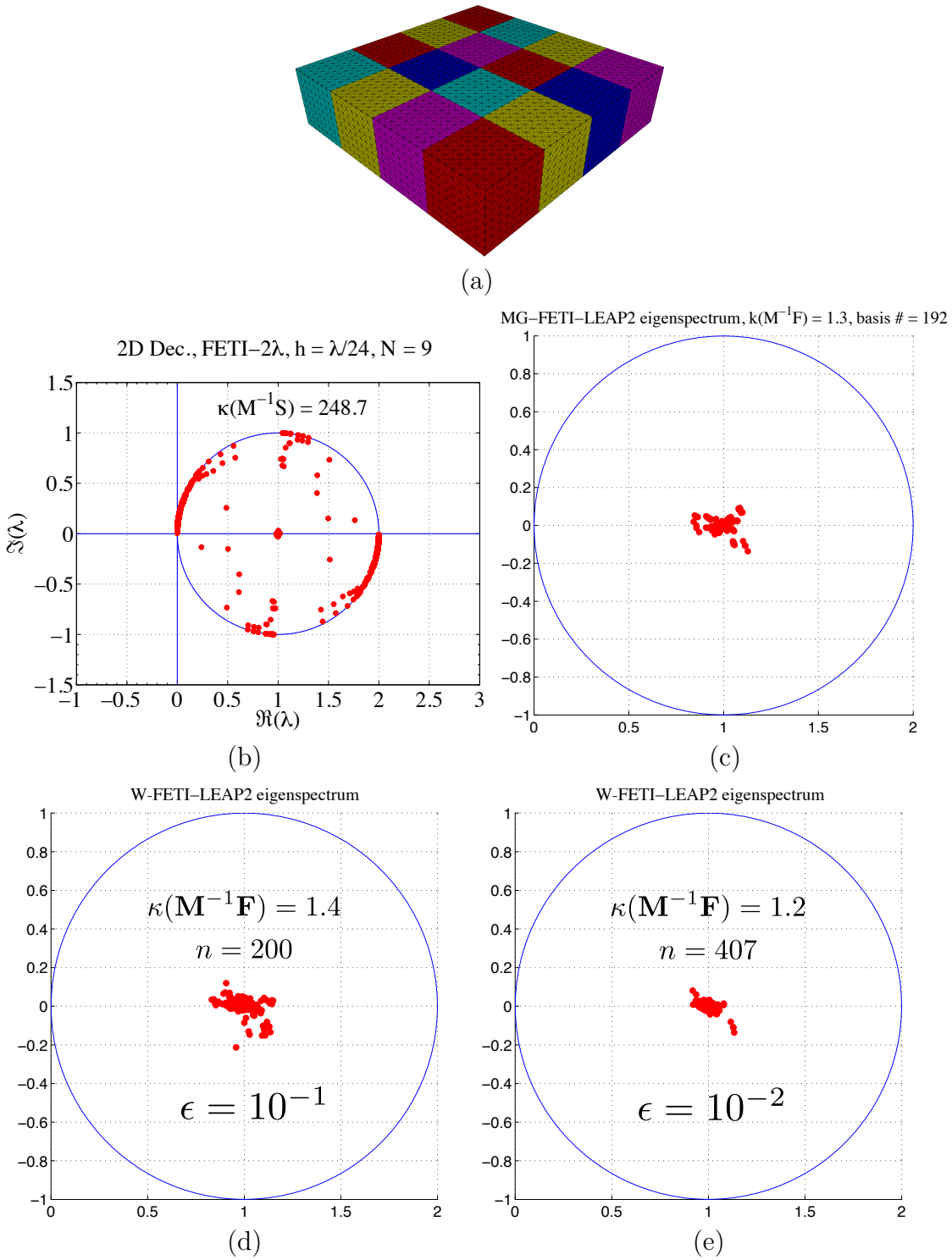


Figure 4.4. A 2D parallel waveguide example with $16(4 \times 4)$ domains: (a) Geometry; (b) Eigenspectrum of the original FETI-2λ matrix; (c) Eigenspectrum with MG-FETI-LEAP2; (d) Eigenspectrum with W-FETI-LEAP2, $\varepsilon_{\text{svd}} = 10^{-1}$; (e) Eigenspectrum with W-FETI-LEAP2, $\varepsilon_{\text{svd}} = 10^{-2}$.

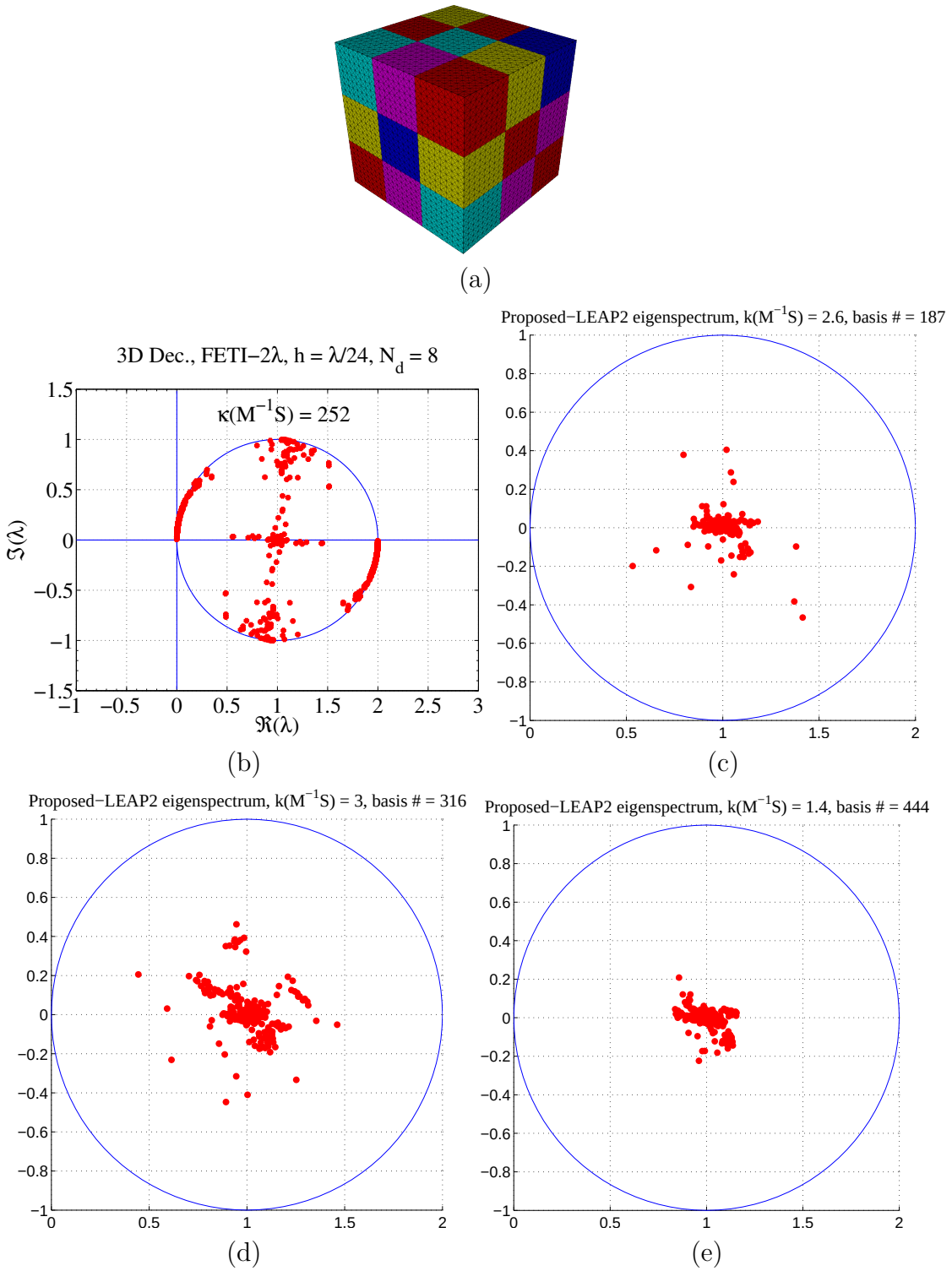


Figure 4.5. A 3D parallel waveguide example with $27(3 \times 3 \times 3)$ domains: (a) Geometry; (b) Eigenspectrum of the original FETI-2λ matrix; (c) Eigenspectrum with MG-FETI-LEAP3; (d) Eigenspectrum with W-FETI-LEAP3, $\varepsilon_{\text{svd}} = 10^{-1}$; (e) Eigenspectrum with W-FETI-LEAP3, $\varepsilon_{\text{svd}} = 10^{-2}$.

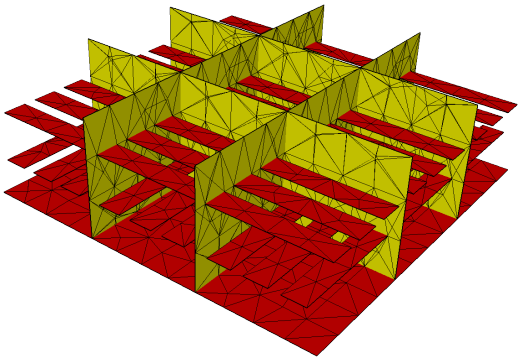
ber from 2.5 to 2.1 while keeping almost the same size of the global preconditioner (n=371, compared to $n = 360$ for MG-FETI-LEAP2).

4.2.2 Structured vs. Unstructured Decomposition

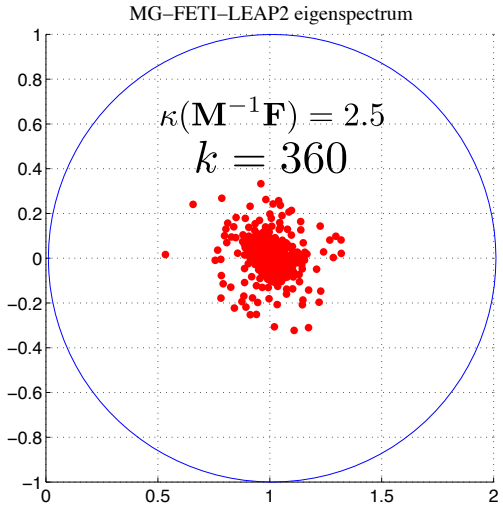
In this section we study the effect of the proposed preconditioner with different decomposition approaches. The same parallel plate waveguide model is used and is decomposed in a 2D topology resulting totally 25(5×5) domains with two different decomposition approaches. In one experiment, the geometry is decomposed in a checker-board structured fashion, where all interfaces are planar, whereas in the other experiment, the mesh of the geometry is decomposed in an unstructured way where the interface are allowed to be arbitrary in shape and are non-planar. While the unstructured decomposition is sometimes more convenient because they can be obtained from partitioning the mesh, it usually results in decompositions with jagged interfaces, leading to worth convergence rate. The results are shown in Fig. 4.7. The mesh from two different decomposition are shown in Fig. 4.7(a)-(b), and the converegence history resulting from the two different meshes is shown in Fig. 4.7(c). It is observed that one more matrix-vector-multiplication is needed to convergece at the same level tolerance with the unstructured mesh, which is expected from previous discussion. It is also pleasant to see the convergence rate does not degrades significantly.

4.2.3 Conforming vs. Non-conforming Decomposition

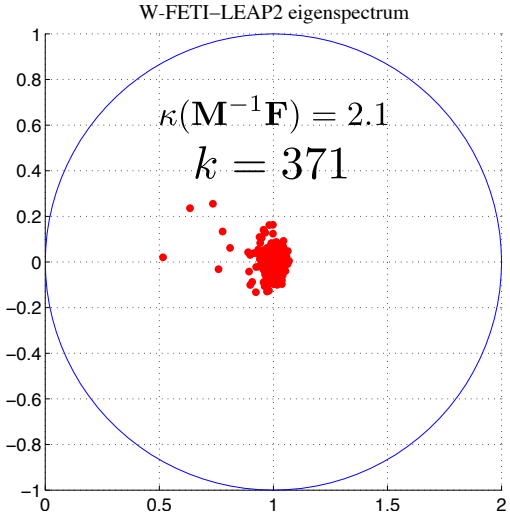
In this section, the W-FETI DDM is tested with conforming and non-conforming (disjoint mesh across interfaces) grids. This is important specifically for multi-scale geometries, because the capability of DDM to work with non-conforming grids enables to mitigate the challenging unstructured meshing burden by allowing independent meshing of domains of the problem. There exist two types of non-conformities in



(a)

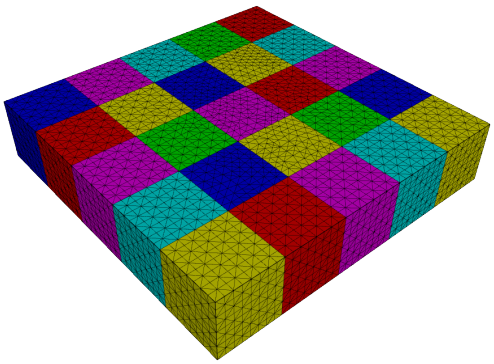


(b)

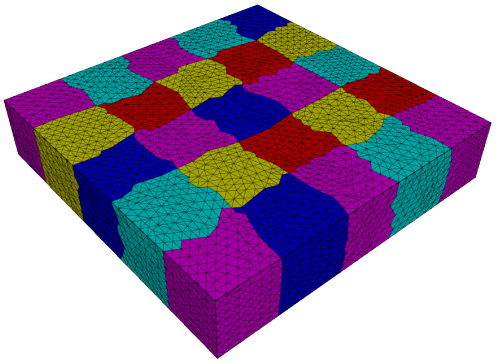


(c)

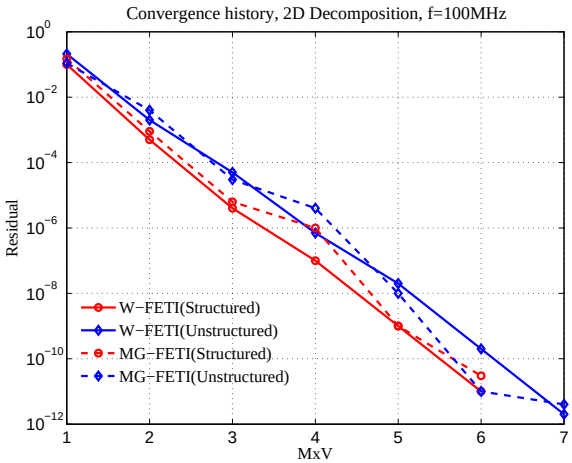
Figure 4.6. A microstrip cross structure model decomposed with 9(3×3) domains; (a) Geometry (red-PEC microstrip, yellow-domain interfaces); (b) Eigenspectrum with MG-FETI-LEAP2; (c) Eigenspectrum with W-FETI-LEAP3, $\varepsilon_{\text{svd}} = 10^{-1}$.



(a)



(b)



(c)

Figure 4.7. A parallel waveguide problem decomposed into $25(5 \times 5)$; (a) Structured decomposition; (b) Unstructured decomposition; (c) Convergence history versus matrix-vector-product (MxV) comparison between MG-FETI-LEAP2 and W-FETI-LEAP2.

DDM, geometrical non-conforming mesh and non-conforming interface mesh. We study both of them here.

We first consider the geometrically non-conforming case. The same parallel waveguide problem decomposed into $16(4 \times 4)$ domains is considered. In this case, the domains are disjointly aligned at interfaces as shown in Fig. 4.8(b) as compared to the conforming one in Fig. 4.8(a). The convergence history of the error residual versus matrix-vector-product count arising from the two different decomposition topologies are shown in Fig. 4.8(c). It indicates the W-FETI DD convergence rate does not degrades when a geometrically non-conforming decomposition is adopted. Next, we test the second type of non-conformity as shown in Fig. 4.9(a) where the domain geometry interfaces perfectly match at interface junction but with non-conforming triangular grids. A pictoral illustration of the non-conforming triangle mesh of one domain interface is plotted in Fig. 4.9(b). The convergence history comparison shown in Fig. 4.9(c) indicates that the convergence rate appears to be identical regardless of the interface conformities.

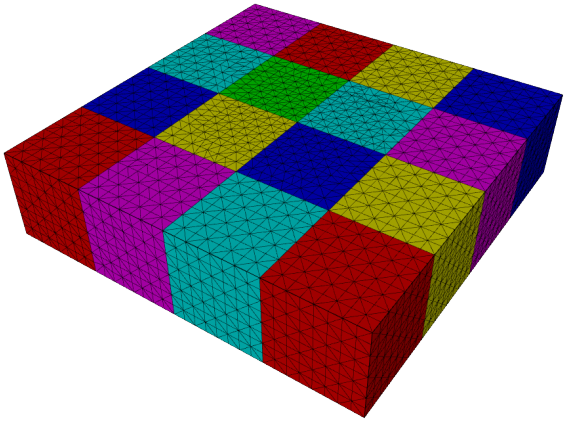
4.2.4 Numerical Scalability Study

In the development of multiscale computational EM methods with DD, the scalability of the method is always the most important benchmark to test. Specifically, ones are interested in the scalability defined as following:

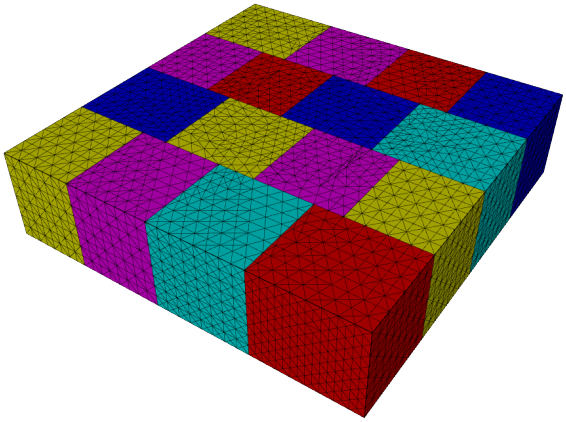
(1) Convergence performance versus domain discretization level and count.

(2) Run time speed-up versus deployed processor/CPU core number.

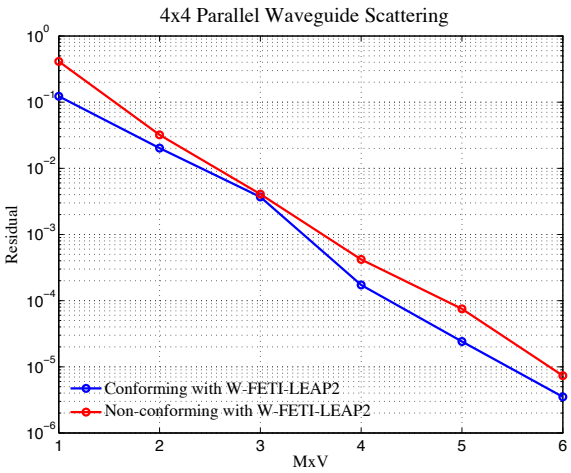
These important scalability benchmarks determine if the developed DD is suited for real-life challenging simulations and state-of-art multi/many-core computing platforms. In this section, we will showcase the achieved scalability performance of the proposed W-FETI DDM for different numerical examples.



(a)

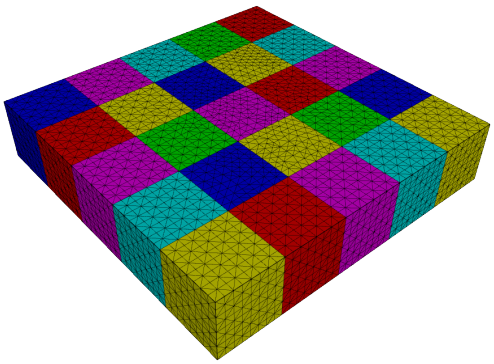


(b)

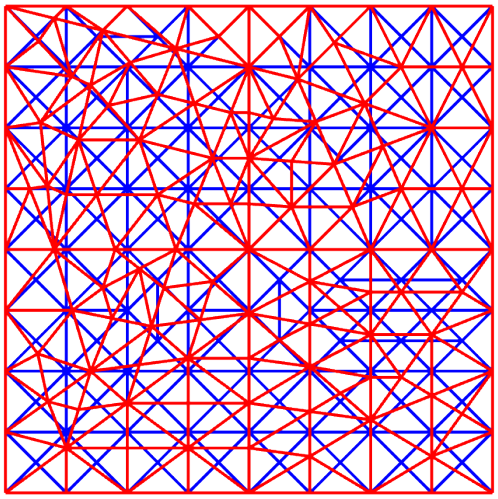


(c)

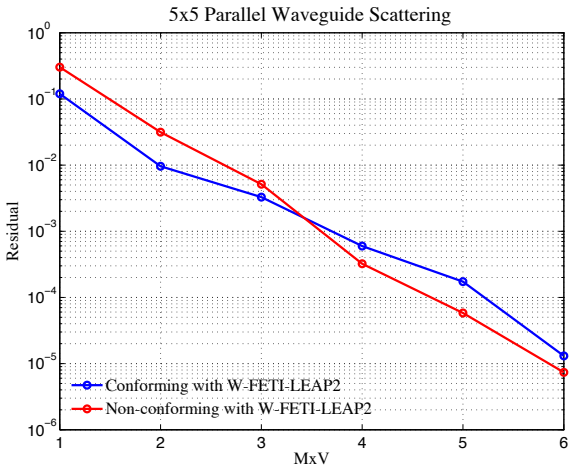
Figure 4.8. A parallel waveguide problem decomposed into $16(4 \times 4)$ to study effects of geometrically non-conforming decomposition; (a) Geometrically conforming decomposition; (b) Geometrically non-conforming decomposition; (c) Convergence history versus matrix-vector-product (MxV) comparison.



(a)



(b)



(c)

Figure 4.9. A parallel waveguide problem decomposed into $25(5 \times 5)$ to study effects of non-conforming mesh decomposition; (a) Geometrically conforming with non-conforming mesh at domain interfaces; (b) Pictoral non-conforming interface mesh [1]; (c) Convergence history versus matrix-vector-product (MxV) comparison.

4.2.4.1 Numerical Scalability w.r.t. Domain Discretization

This section investigates the numerical scalability of W-FETI w.r.t. domain discretization. A parallel waveguide problem with 9 domains in 2D decomposition topology is considered. The discretization level at each domain starts from $h = \lambda/8$ up to $h = \lambda/48$. The problem is solved at $f = 50MHz$. Fig. 4.10(a)-(b) show a comparison of the mesh discretization at $h = \lambda/8$ and $\lambda/16$. The iterative convergence versus matrix-vector-product of W-FETI at different discretization level are plotted in Fig. 4.10(c). As can be seen, the four runs converge with identical number of MxV indicating W-FETI is scalable w.r.t. domain discretization. To compare with MG-FETI, the scalability versus total problem unknown size are shown in Fig. 4.10(d). It can be concluded that both W-FETI and MG-FETI are scalable w.r.t. domain discretization h. Computational statistics of this experiment are provided in Table 4.2.

Table 4.2. Computational statistics of the domain discretization scalability experiment conducted on a 2D parallel waveguide problem.

# Discretization h	Method	# Unknowns (K)	# MxV	Time (h:m:s)	Memory [MB]
$\lambda/8$	MG-FETI	32	7	00:01:12	89
	W-FETI	32	6	00:01:10	87
$\lambda/16$	MG-FETI	218	7	00:03:59	489
	W-FETI	218	6	00:03:56	485
$\lambda/32$	MG-FETI	1725	7	00:48:32	4,201
	W-FETI	1725	6	00:47:11	4,195
$\lambda/48$	MG-FETI	4153	7	01:50:12	19,054
	W-FETI	4153	6	01:48:44	18,892

4.2.4.2 Numerical Scalability w.r.t. Domain Count

In this section, an EM scattering problem due to an incident wave propagating along z-axis and polarized along y-axis to a PEC thin plate is considered. The geometry of the smallest plate and its decomposition is shown in Fig. 4.11(a)-(b). All domains are cubes with $\lambda \times \lambda \times \lambda$ size at 50 MHz and discretized with about 45,000

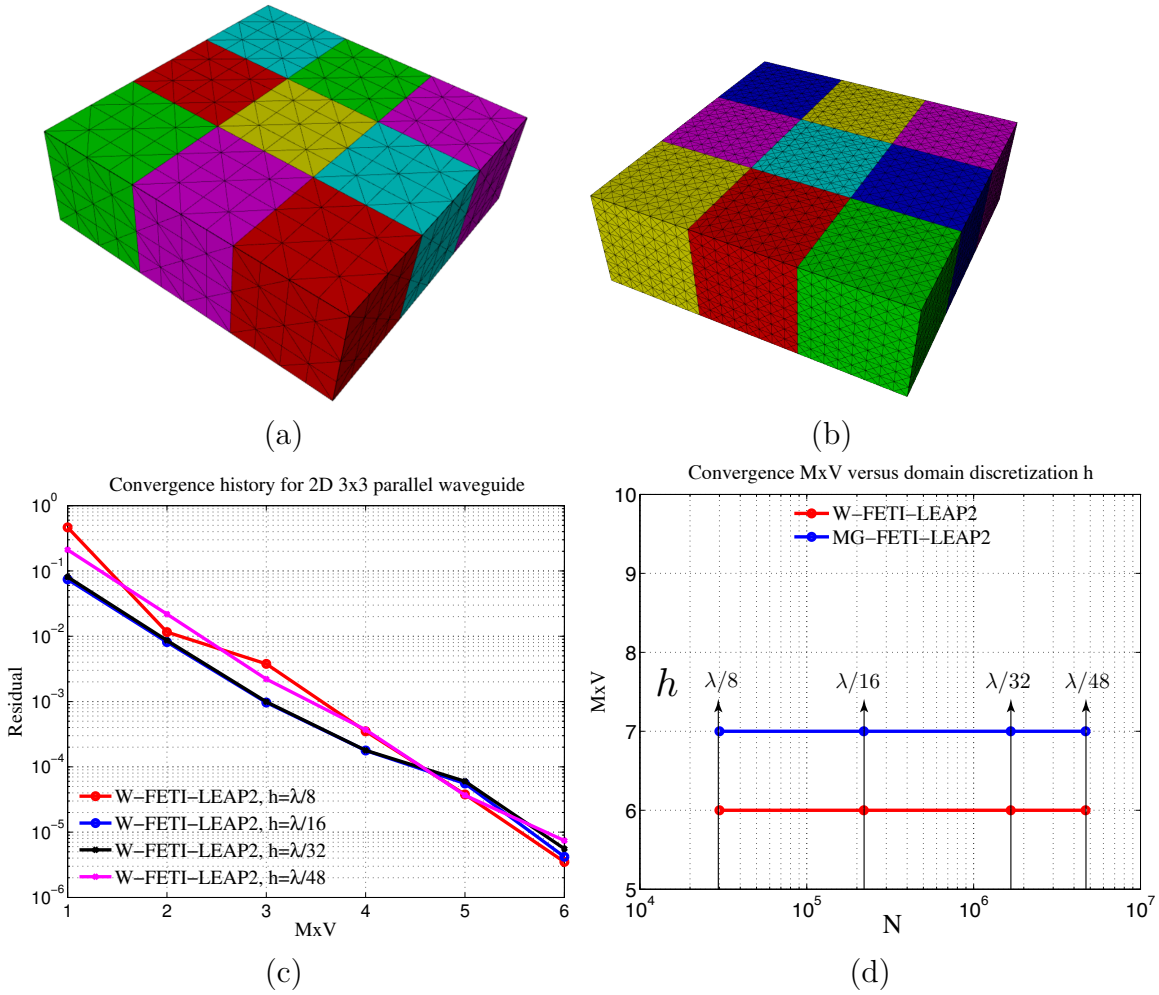


Figure 4.10. A parallel waveguide problem with 9 domains to study the scalability w.r.t. domain discretization; (a) Mesh of the problem at discretization level $h = \lambda/8$; (b) Mesh of the problem at discretization level $h = \lambda/16$; (c) Convergence comparison of W-FETI at different discretization level; (d) Scalability comparison w.r.t. domain discretization between W-FETI and MG-FETI.

($h = \lambda/6$) second-order FEM unknowns. The computational problem is decomposed in a 2D topology, and it becomes electrically larger by increasing the domain count.

In this experiment, the W-FETI SVD tolerance is set to $\varepsilon_{\text{svd}} = 10^{-1}$. Fig. 4.11(c) show the convergence comparison between the W-FETI, MG-FETI and FETI-DP preconditioned FETI-2λ system for the same PEC plate scattering problem with 121 decomposed domains. The proposed W-FETI-LEAP2 preconditioning provides the best convergence rate among the three global preconditioning methods. In Fig. 4.11(d), the PEC plate size is progressively increased from $\lambda \times \lambda$ to $9\lambda \times 9\lambda$ and the iterative convergence performance is compared between W-FETI and MG-FETI with decompositions of different domain count. It is observed that the convergence rate of W-FETI remains almost identical for all test cases whereas the opposite for MG-FETI. The scalability plot of number of matrix-vector-product needed to converge at 10^{-6} versus domain count is shown in Fig. 4.11(e). It clearly indicates that the W-FETI preconditioner offers a stable iteration number required to converge for different domain count, while the iteration number arising from MG-FETI approach is 16 at the smallest electrical size, and increases up to 26 at the largest electrical size. FETI-DP showcases the worst scalability among the three. This suggests that the proposed W-FETI preconditioner is scalable w.r.t. domain count and unknown size. The computational statistics with all the detailed description on the memory and unknown size is provided in Table 4.3.

4.2.4.3 Parallel Scalability

The parallel scalability of the proposed DD framework is studied in two different examples. Before introducing the details of the numerical experiments, it is benefical to rigorously define the parallel scalability. In this work, we use parallel speed-up an parallel efficiency,

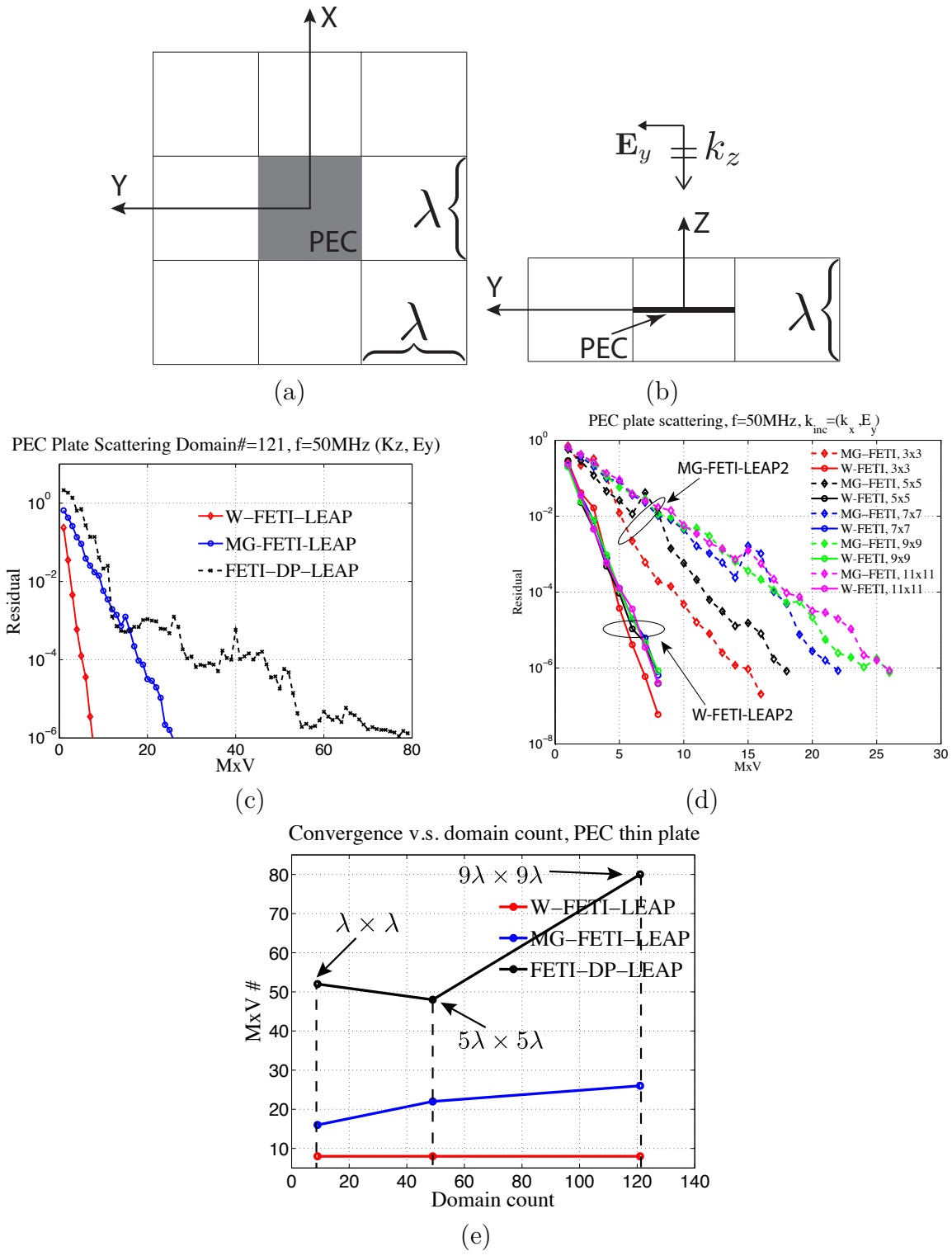


Figure 4.11. A PEC thin plate scattering problem to study the scalability w.r.t. domain count; (a) Top view of the problem with the smallest domain count($N = 9$); (b) Side view; (c) Convergence comparison of W-FETI, MG-FETI and FETI-DP of the problem with the largest domain count($N = 121$); (d) Convergence comparison of W-FETI and MG-FETI with different domain count; (e) Number of matrix-vector-product vs. domain count.

Table 4.3. Computational statistics of the domain count scalability experiment conducted on a 2D PEC plate scattering problem.

# Domain	Method	# Unknowns (K)	# MxV	Time (h:m:s)	Memory [GB]
9	MG-FETI	425	16	00:09:15	0.8
	W-FETI	425	8	00:08:58	0.78
25	MG-FETI	1,161	18	00:23:42	2.5
	W-FETI	1,161	8	00:21:18	2.5
49	MG-FETI	2,258	22	00:48:17	5.51
	W-FETI	2,258	8	00:45:19	5.50
81	MG-FETI	3,744	26	01:25:08	9.62
	W-FETI	3,744	8	01:21:48	9.59
121	MG-FETI	5,617	26	02:10:31	14.63
	W-FETI	5,617	8	02:06:41	14.59

$$\text{speed-up} = \frac{T_{\text{S}}}{T_{\text{P}}} \tag{4.19}$$

$$\text{efficiency} = \frac{T_{\text{S}}/T_{\text{P}}}{N_{\text{core}}} \tag{4.20}$$

where T_{S} is the "old" execution time, often is identical to $T_{N_{\text{core}=1}}$, T_{P} is the parallel execution time.

The discussion in the previous chapter suggests that the computation of domain DtN map operators plays a dominant role in computational resource overhead both in run-time and memory, therefore the DtN computation stage has a major effect on scalability. In order to benchmark the parallel scalability of W-FETI global preconditioner, work load to compute domain DtN matrices on each CPU core need to be almost perfectly balanced. To achieve this, a simple quadratic regression model based on z-matrix size to approximate its computational overhead is designed and a greedy domain distribution algorithm is adopted to distribute work load to each CPU core. The model is constructed by utilizing historical computational statistics for domain DtN matrices, as shown in Fig. 4.12. It should be emphasized that the constructed model needs to be updated with new data when it is available.

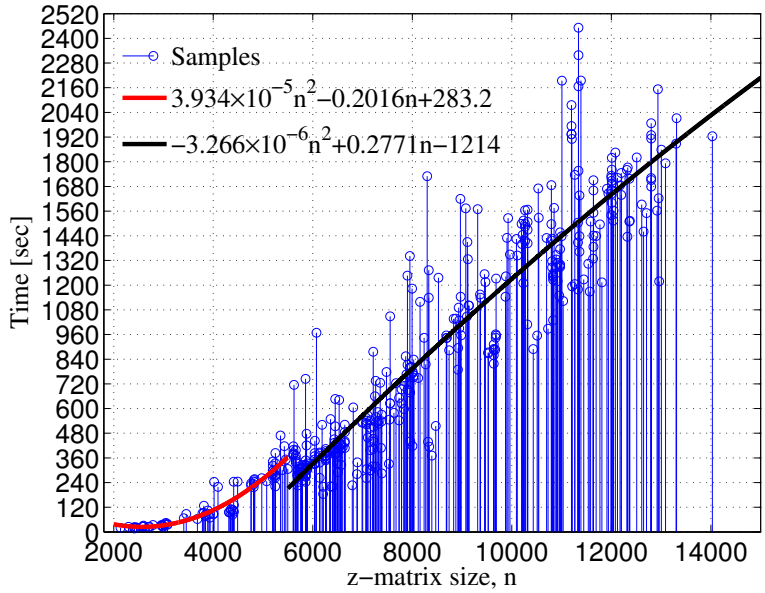


Figure 4.12. A quadratic model to approximate time and memory cost of computing domain DtN map operator based on its size.

We first consider a PEC cavity scattering problem due to an incident wave along its longitudinal axis at $f = 100$ MHz. As shown in Fig. 4.13(a), the model is decomposed into 16 domains with a 3D decomposition topology. The problem is discretized with a total number of 382,663 tetrahedrons, resulting 2,420,528 second-order FEM unknowns. Fig. 4.13(b) shows the iterative convergence comparison to an error tolerance $\varepsilon = 10^{-6}$ with W-FETI-LEAP3 and MG-FETI-LEAP3. It is observed that W-FETI-LEAP3 takes less iterative passes to reach the desired error residual. Fig. 4.13(c)-(d) show the computational overhead of z-matrices in time and memory at each CPU core when a total number of 8 CPU cores are used for the parallel run. It can be seen that the empirical model combining with the greedy distribution algorithm provides a good load balance when $N_{\text{core}} = 8$. The achieved parallel speed-up versus number of cores is plotted in Fig. 4.13(e) whereas the corresponding parallel efficiency is shown in Fig. 4.13(f). It is expected to see the speed-up/efficiency degrades by increasing the number of cores as the computational overhead of z-matrix for each domain is not the same. The proposed W-FETI-LEAP3 DD achieves roughly 78%

parallel efficiency at $N_{\text{core}} = 16$ which is considered to be considerably good for a large-scale DD run.

Next a scattering problem of a free-space cube that is decomposed into 160 domains with 8,610,583 second-order FEM unknowns is considered. With 80 CPU cores, an almost 80% parallel efficiency is achieved as shown in Fig. 4.14. It is worth to point out that the parallel efficiency is expected to considerably degrade when a more realistic example is tested as the z-matrix load balance problem becomes more profound, interested readers can refer to the next chapter for more details.

4.3 Discussion

A effective and robust global preconditioning technique is introduced in this chapter. The proposed W-FETI global preconditioner is algebraic and scalable w.r.t. domain discretization and count. The combination of W-FETI and LEAP^p provides a very robust preconditioning methodology, numerical results indicate that W-FETI appears to perform at least as robust and effective as two state-of-art global preconditioners, the MG-FETI and FETI-DP, and some challenging cases show that the W-FETI outperform its competitors.

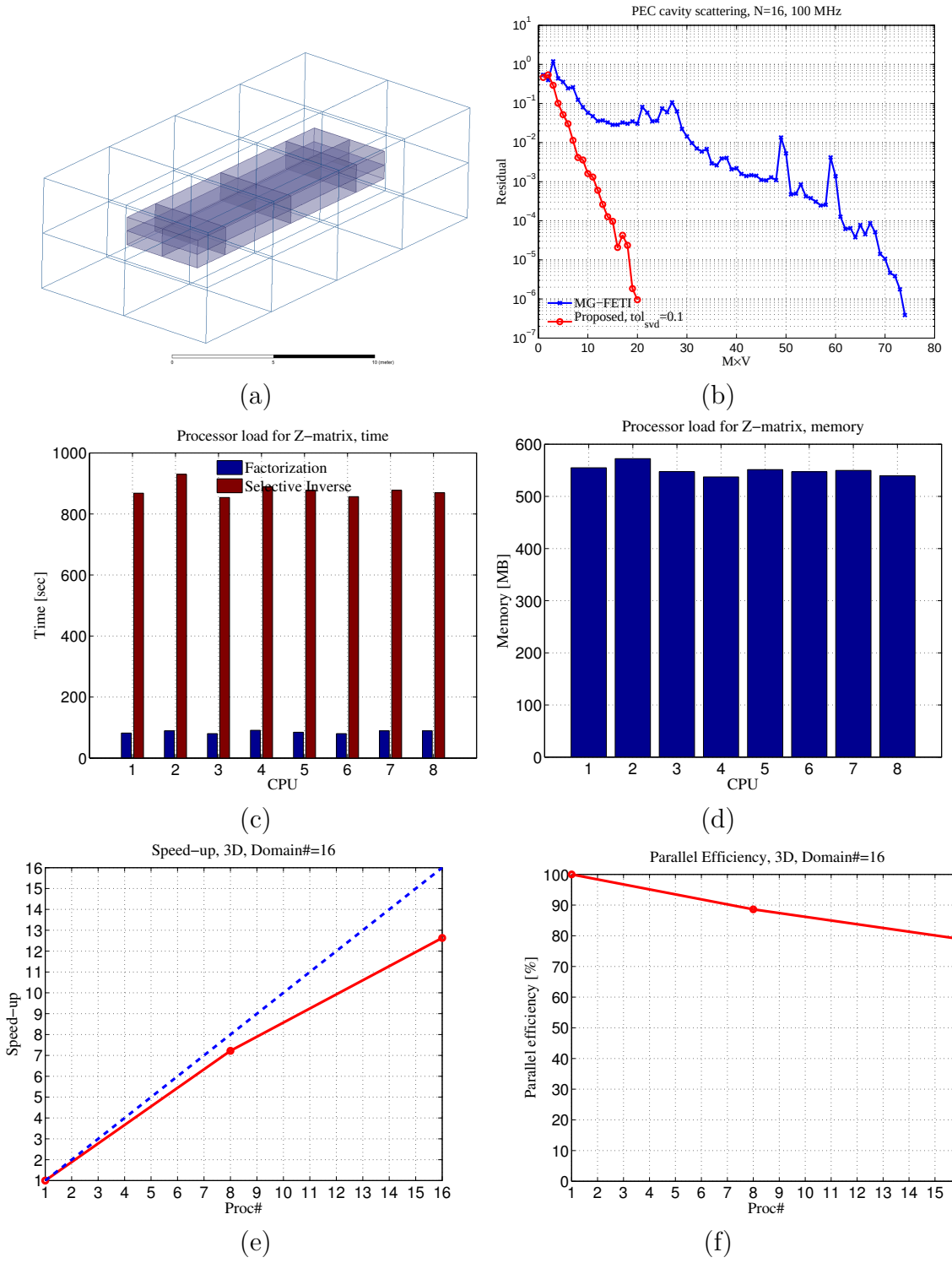


Figure 4.13. A PEC cavity free-space scattering problem decomposed in 3D topology with 16 domains to study the parallel scalability of W-FETI; (a) Model view with domain interfaces; (b) Iterative convergence comparison between W-FETI and MG-FETI; (c) Run-time of domain DtN map operators at each CPU core($N_{\text{core}} = 8$); (d) Memory cost of domain DtN map operators at each CPU core($N_{\text{core}} = 8$); (e) Parallel speed-up versus number of CPU core; (f) Parallel efficiency versus number of CPU core.

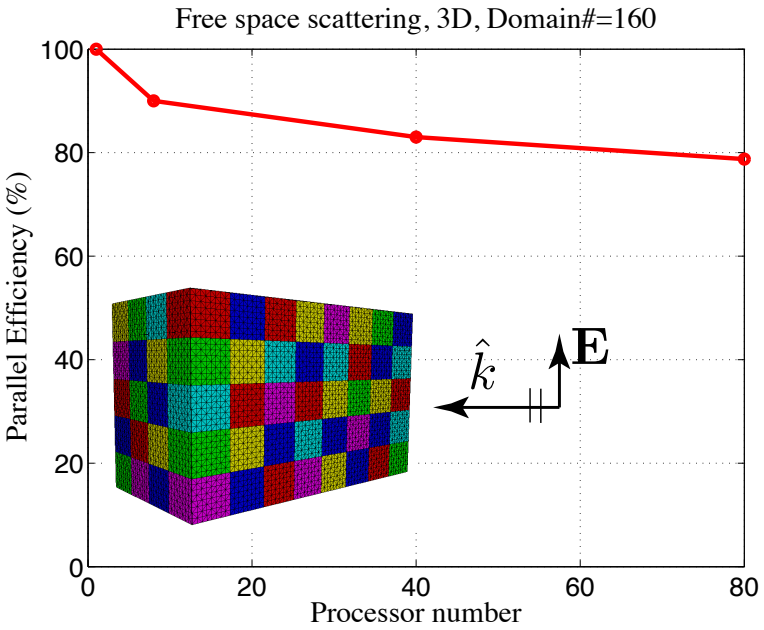


Figure 4.14. Achieved parallel efficiency for simulating a free-space scattering problem decomposed in 3D topology with $160(4 \times 5 \times 8)$ domains.

CHAPTER 5

NUMERICAL RESULTS

Numerical experiment results from previous chapters suggest a promising outlook for the W-FETI accelerated by the randomized DtN computation. In this chapter, we further test experimentally this outlook by testing some challenging and industrial graded examples and document and compare the performance with state-of-the-art competing CEM technologies. These test examples are drawn from several diverse electrical engineering disciplines, including microwaves, signal integrity and electromagnetic compatibility, antennas and scattering. More specifically we consider the transmission and reflection from a X-band waveguide filter, signal propagation along traces of multi-layer printed circuit board, the signal integrity of an integrated circuit (IC) package, radiation by a finite Vivaldi array enclosed with a radome and a TEM wave scattering from a generic drone aircraft. These examples have electrical sizes that range from deep-sub-wavelength, in the case of the the IC package and PCBs,to near resonant, in the case of the waveguide filter, to electrically large, 40 linear wavelengths, in the case of the drone aircraft.

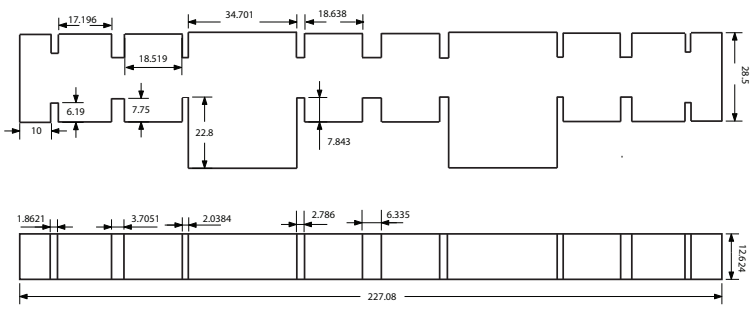
All the computations were performed with DDM based on first kind Nedelec ($p = 1$) tangential vector finite element method (TV-FEM) [75] discretizing with the tetrahedron that requires 20 DoFs per element. The simulation setup and parameters, software development and hardware paltforms used for all these runs remain the same for all these test cases and are introduced below. All floating point operations are in double precision complex arithmetics. The IDR(1) iterative solver [65, 66] is used to solve the non-symmetric FETI-2λ matrix that requires two matrix-vector-products

(MxVs) per iteration. The error tolerance for adaptive randomized computation for domain discrete DtN and fast SVD computation in W-FETI `tol`$= 10^{-1}$ is used unless explicitly stated otherwise. The SVD truncation tolerance defined in W-FETI is set as $\varepsilon_{\text{svd}} = 10^{-2}$ unless explicitly stated otherwise. The code was developed in C++ programming language and compiled with Intel Compilers (v11.1) with -O3 optimization turned on. The adopted parallel computation API is Open MPI v1.4 [76]. The numerical linear algebra packages used are Intel Mathematical Kernel Library (MKL) (v11.1) [72] and MUMPS (v4.10.0) [67]. For serial runs, the simulations are performed on a MacPro workstation with two 2.8 GHz Intel Xeon quad-core processors with 6144KB L2 cache and 32 GB RAM. For parallel runs, they are conducted on a in-hourse developed cluster with 10 MacPro workstations connected through a gigabyte ethernet network, each workstation has two 2.8 GHz Intel Xeon quad-core processors with 6144KB L2 cache and 32 GB RAM.

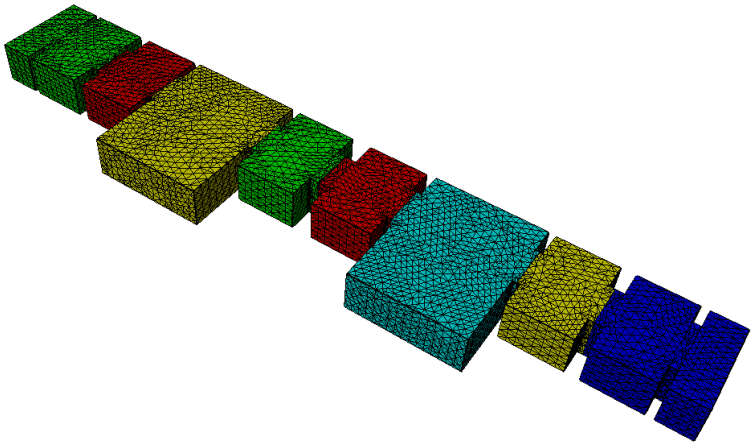
5.1 Waveguide Filter

The first example considered is a band pass microwave waveguide filter with dual mode cavities designed to operate in the X-band [77]. The model has two waveports at its two ends and several perfect electric conductor (PEC) stepped waveguide section and and two cavities, as shown in Fig. 5.1(a)-(b) with details of the model dimensions in milimeters. This problem is relatively simple, but it does poses some challenges to FEM solvers because of the singular fields that develop at the re-entering PEC corners, and the development of higher-order modes withing filter sections. The frequency band of interest is 7.8-9 GHz. The problem is decomposed with one-way (1D) decomposition resulting in 8 domains that is shown in Fig. 5.1(b). The two-port device is discretized using 182,759 tetrahedral elements resulting 1,127,514 second-order TV-FEM primal unknowns and 7,732 dual unknowns from FETI. To verify the accuracy of the proposed DDM, it is compared with the state-of-art multigrid

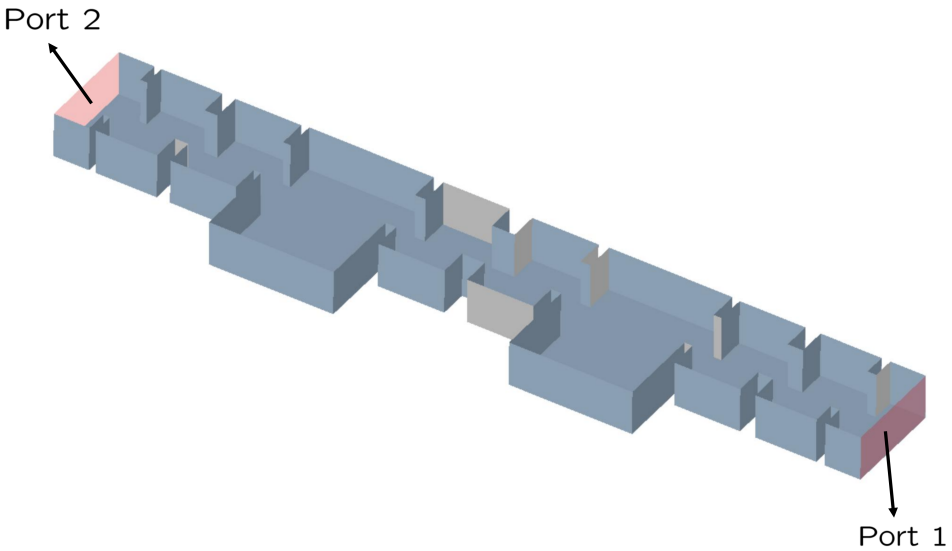
FEM (MG-FEM) method that solves the same model without domain decomposition [13]. A BiCGSTAB(1) solver [78] is adopted for the MG-FEM method. Fig. 5.2(a)-(d) show the s-parameter amplitude and its phase comparison with MG-FEM over the entire frequency band. The W-FETI frequency response well agress with the MG-FEM s-parameter results. Fig. 5.2(e)-(f) plot the corresponding relative error $|s - s_{\text{ref}}|/|s_{\text{ref}}|$ showing that both error stay below -25dB at all simulated frequencies. These errors are primarily attributed to the randomized compression of the discrete DtN operator, that in this case are computed with tolerance $\texttt{tol} = 10^{-}2$. These errors were shown to be controllable in the Chapter 3, merely by reducing this tolerance, at the cost of extra computational resources. It can be concluded that W-FETI is as accurate compared to the well-established MG-FEM. The convergence history of the error residual versus matrix-vector-product and time are given in Fig. 5.3(a)-(b). W-FETI reaches the desired residual $\varepsilon = 10^{-10}$ with 5 matrix-vector-products whereas MG-FETI-LEAP needs 6 matrix-vector-products. To give a idea of the savings attributed to the randomized low-rank approximation of the discrete DtN operations, the total time is reduced by 22%, and the memory savings are reaching 27%. Detailed computational statistics for the waveguide filter problem are given in Table 5.1. It is worth noting that for this example the proposed method uses approximately five times less memory than non-domain decomposition state-of-the-art solvers, but at the expense of computational time. This additional time overhead could be remedied by using parallel computations as it will be shown later in this chapter.



(a)



(b)



(c)

Figure 5.1. A X-band waveguide filter model and its one-way decomposition with 8 domains: (a) Dimension details of the model; (b) Model view; (c) Decomposition layout view.

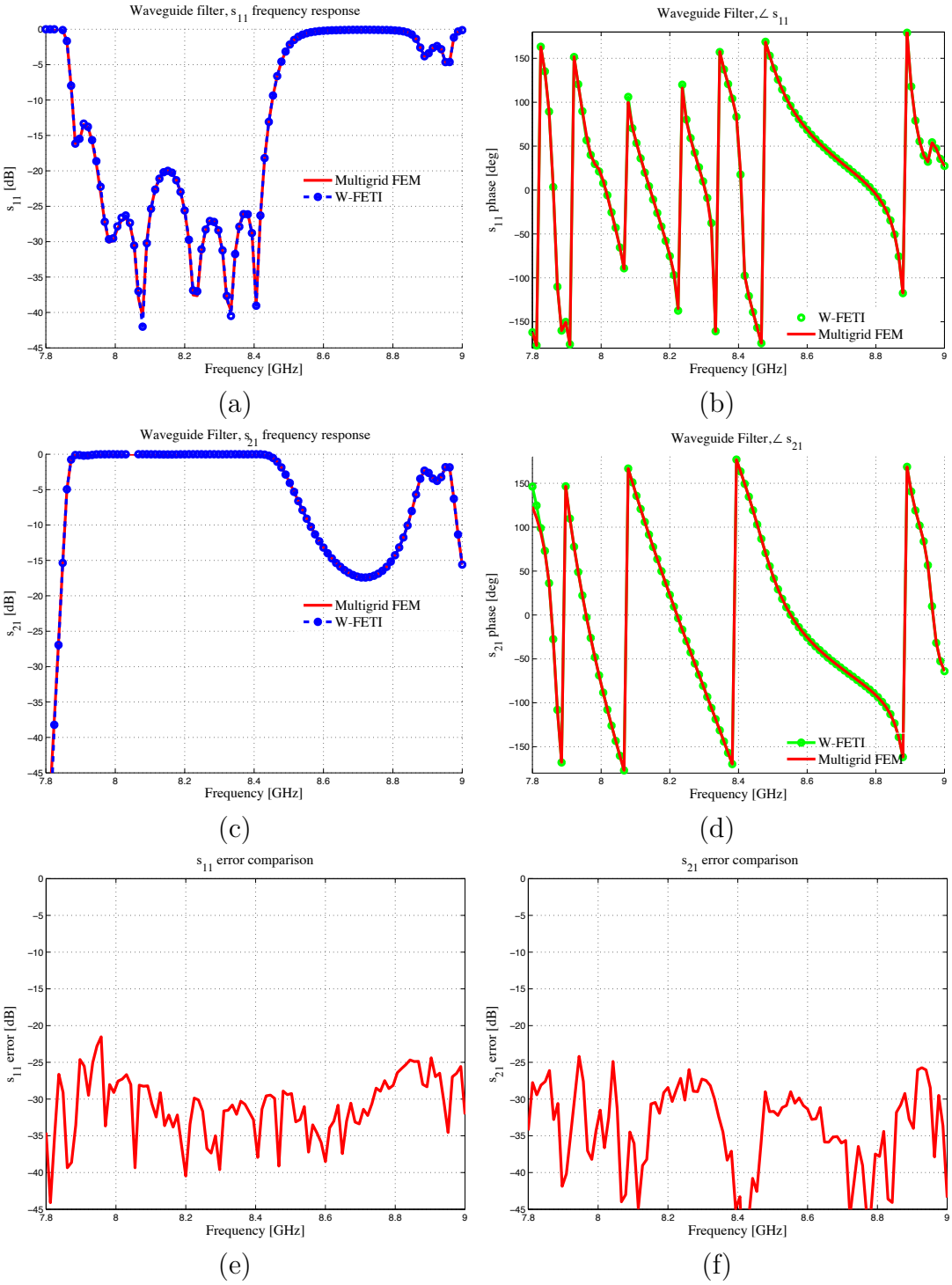
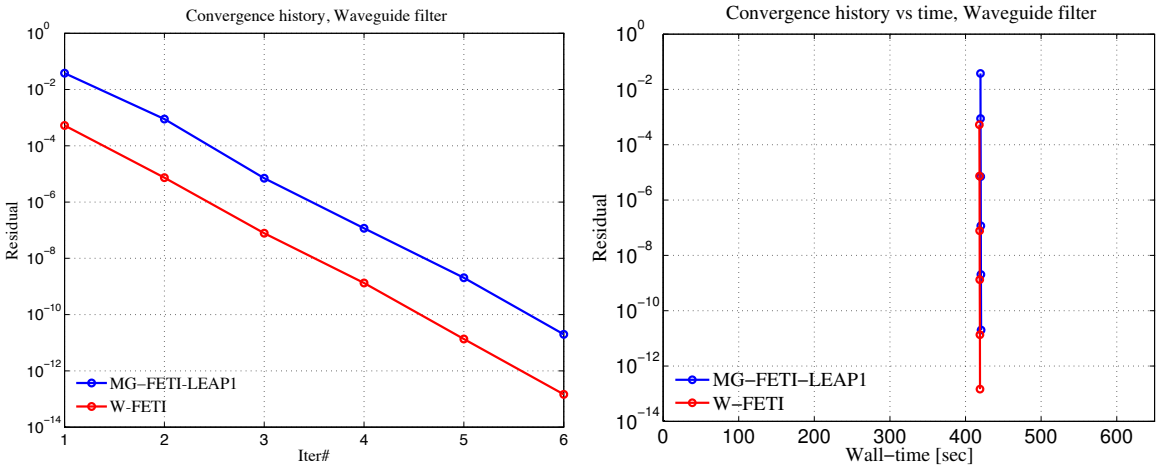


Figure 5.2. s-parameter comparison with MG-FEM for the waveguide filter example: (a) $|s_{11}|$; (b) Phase of s_{11}; (c) $|s_{21}|$; (d) Phase of s_{21}; (e) Error of s_{11}; (f) Error of s_{21}.

Table 5.1. Computational Statistics on a waveguide filter problem with one-way (1D) decomposition.

Domain count	Method	Unknown number (K)	# of MxVs (tol=10^{-10})	Time (hh:mm:ss)	Memory [MB]
8	W-FETI[a]	1,120	5	00:05:18	272
8	MG-FETI [1]	1,120	6	00:06:48	377
1	MG-FEM [13]	901	91	00:03:08	1541

[a]With compressed representation of domain discrete DtN oeprators using randomized algorithms.



(a) (b)

Figure 5.3. Iterative convergence history of the W-FETI and MG-FETI-LEAP for the waveguide filter problem: (a) Iterative convergence history versus matrix-vector-product; (b) Iterative convergence versus wall-time.

5.2 Signal Integrity Analysis of Multilayer PCB

This section considers the same printed circuit board as shown in the Chapter 3 result section. We focus on studying the W-FETI performance and parallel efficiency here. The model is 110×70 [mm] and approximately 5,000 vias are placed across the layers which further increases the geometry complexity. The problem is discretized with 1,234,054 tetrahedrons resulting a total number of 7,110,100 TV-FEM primal unknowns and 115,146 dual unknowns. The entire model is decomposed into 15 domains with a 2D partition topology. It should be emphasized that the decompositions are by no means optimal, in fact the domain interfaces have been intentionally chosen to occur at regions of strong singular and evanescent fields, and to maximize the number of dual unknowns simply to test the resilience and reliability of the proposed methods. Fig. 5.4(a) shows the convergence history of the residual versus matrix-vector-product for W-FETI for the Woodbury SVD truncation $\varepsilon_{\text{svd}} = 0.1$ and 0.01 compared with MG-FETI-LEAP2, FETI-DP and MG-FEM at $f = 2.5$ GHz. This operating frequency is chosen by intention because it was reported in [1] that the MG-FETI-LEAP and other methods have the most difficulty to converge at this frequency. Minimum LEAP2 buffer region is adopted in this simulation, as shown in Fig. 5.5(a). It clearly shows that W-FETI is the only method that can convergence to a error residual of 10^{-6}. When $\varepsilon_{\text{svd}} = 0.01$ is used, W-FETI converges extremely fast in only 9 matrix-vector-products. The W-FETI global matrix size is $k = 591$ when $\varepsilon_{\text{svd}} = 0.1$ and $k = 1611$ when $\varepsilon_{\text{svd}} = 0.2$. The residual versus run-time is plotted in Fig. 5.4(b) with both serial and parallel ($N_{\text{core}} = 15$) runs. Several different runs with different nubmer of CPU cores are also performed to document the parallel efficient is plotted in Fig. 5.4(c). It should be emphasized that this particular decomposition is rather unbalanced, i.e. the number of primal and dual unknowns between domains varies significantly, leading to rather low parallel efficiencies for the cases

with large number of CPU resources. The computational statistics of this simulation is provided in the first half section of Table 5.2.

As shown in [1], the size of the buffer region of LEAP2 is an user defined input parameter of MG-FETI-LEAP method that critically affects the iterative convergence and ultimately the speed, memory and reliability for the success of MG-FETI-LEAP. To demonstrate the resilience and relative insensitiveness of the proposed W-FETI method on this somewhat arbitrary user-defined parameter, three different buffer sizes are used to solve the example and computational model described above. Fig. 5.5(a)-(c) show the adopted buffer region from its minimum size to the maximum. In this study, $\varepsilon_{\text{svd}} = 0.1$ is used. Fig. 5.5(d) plots the convergence comparison of W-FETI and MG-FETI with the aforementioned LEAP2 buffers. For the minimum buffer, W-FETI managed to converge to the residual error of 10^{-6} with 598 matrix-vector-products whereas MG-FETI-LEAP2 fails to converge. It is also noted that the asymptotic convergence slope for W-FETI in all cases are almost identical. These observations indicate that the proposed W-FETI is less sensitive to the user defined buffer size than MG-FETI-LEAP, thus it is more resilient. Even in cases where rather large buffer is used, the W-FETI approach converges in less than half of matrix-vector-products than MG-FETI-LEAP. The benefits by using a smaller buffer region though is in the significant savings in time and memory attributed to the more efficient assembling and storing of the local preconditioners. Detailed computational statistics of these comparison can be found in Table 5.3. It is observed that using the minimum buffer size saves almost 97% in total memory, as compared to the largest buffer case.

It is important to realize that the convergence performance of domain decomposition methods degrades a FETI DD method tends to degrades when the same problem is parittioned into more domains. This is because information is communicated only between neighboring domains in each domain decomposition iteration, and that is why a global preconditioner is necessary and critical to the numerical scalability and

reliability. Contrary, decompositions with large number of domains can use more CPU core resources, but also give more flexibility for load balancing in parallel runs, thus are preferable. In the next computational experiment, the same PCB model is decomposed into 40 domains, resulting a total number of 7.21 million second-order FEM unknowns as shown in the insert of Fig. 5.6(c). Fig. 5.6(a) shows the iterative convergence performance of W-FETI and MG-FETI-LEAP. The proposed W-FETI converges with 26 matrix-vector-products to 10^{-6} whereas MG-FETI-LEAP2 again fails to converge with the same input parameters (i.e., LEAP2 buffer size, etc.). In this run, a moderate size of LEAP2 buffer shown in Fig. 5.5(b) is used. The iterative convergence history versus wall-time is plotted in Fig. 5.6(b) with both serial and parallel runs with 40 CPU cores to show the speed and parallel performance of the method. The achieved parallel efficiency for the 40-domain partitioned model is plotted in Fig. 5.6(c). Finally, Fig. 5.6.(d) shows the updated CPU core usage (load balancing) of this new decomposition model for 8 cores, showing an excellent load-balancing. It is noted that only the discrete DtN computation is shown, since it is the most time consuming part of the computations. It is expected that as the number of CPU cores increases the load-balances quality decreases. A 70% parallel efficiency is achieved when 16 CPU cores are used, which is considerably good for a challenging nature of example. A better parallel efficiency is achieved compared to the 15-domain partition case due to a more balanced load is achieved with the empirical model, as presented in Chapter 4. The detailed computational statistics of this experiment are reported in the bottom half of Table 5.2.

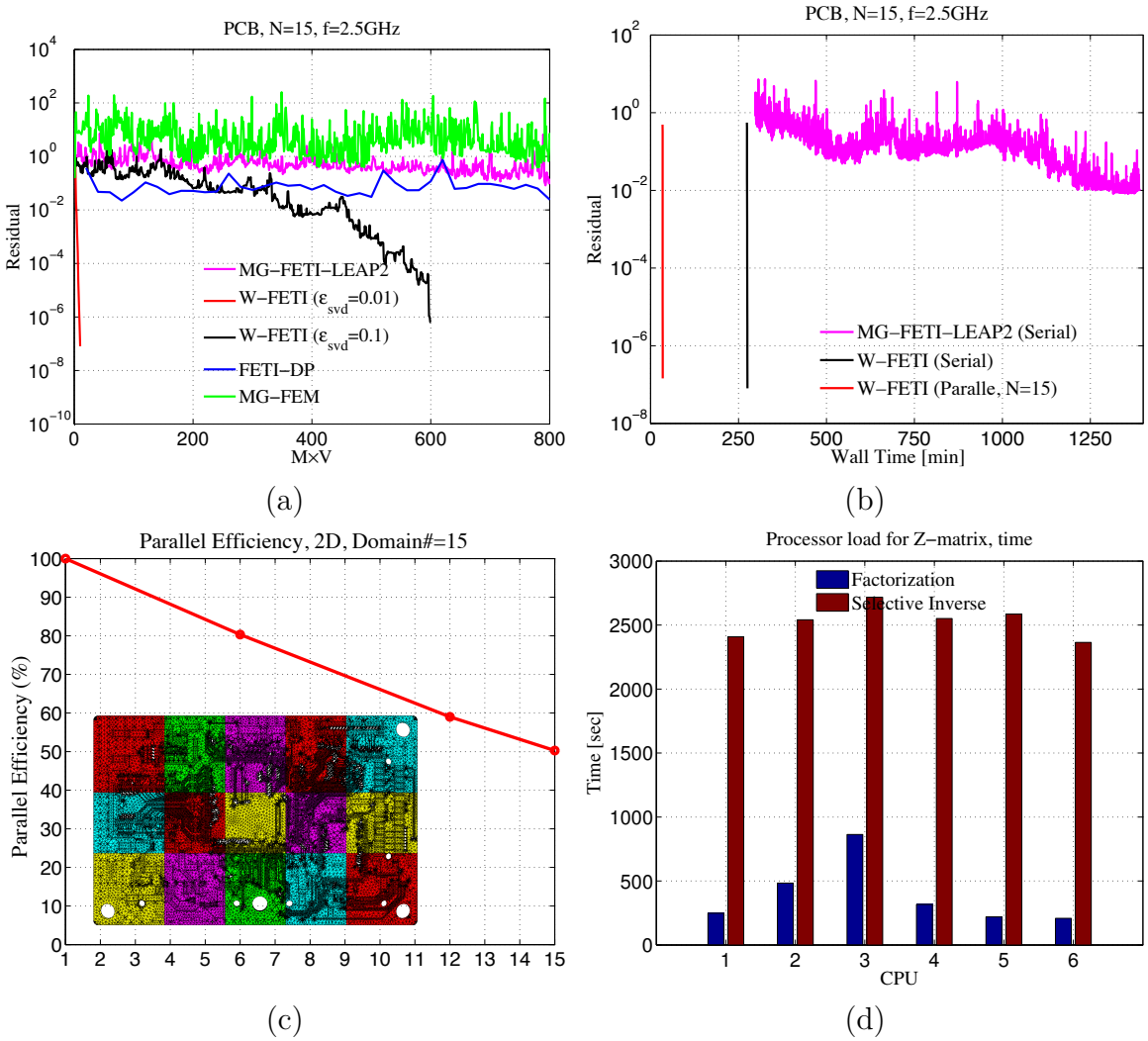
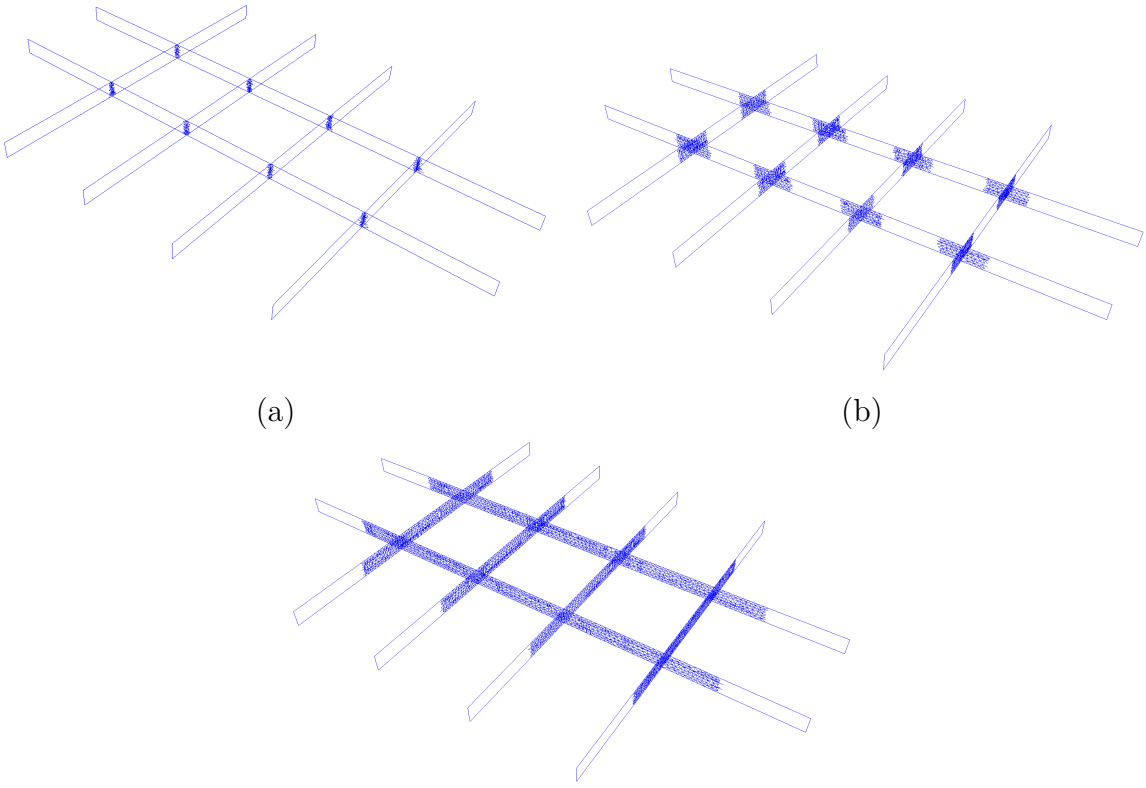
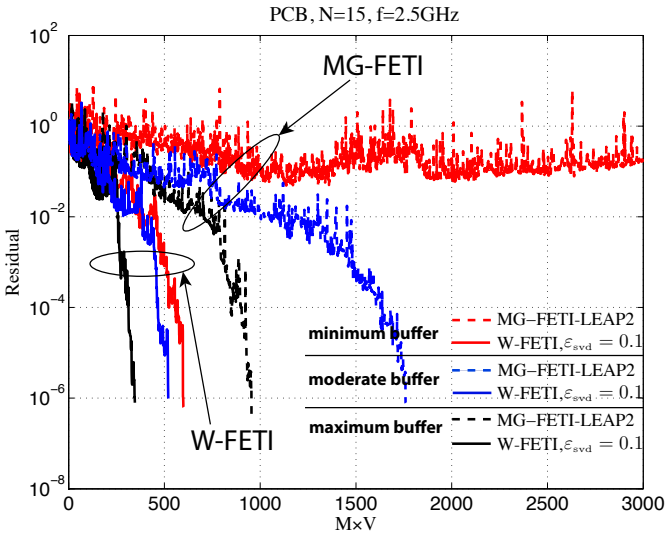


Figure 5.4. Iterative convergence and parallel efficiency the PCB SI analysis example decomposed into 15 domains: (a) Iterative convergence history comparison between W-FETI with different ε_{svd}, MG-FETI, FETI-DP and MG-FEM; (b) Convergence history versus wall-time of W-FETI (serial and parallel) and MG-FETI-LEAP2; (c) Parallel efficiency versus number of CPU cores. (d) Computational time overhead for domain discrete DtN at each CPU core ($N_{\text{core}} = 6$).



(a) (b)

(c)



(d)

Figure 5.5. Effect of LEAP2 buffer size on the iterative convergence of the multi-layer PCB problem: (a) Minimum size; (b) Moderate size; (c) Maximum size; (d) Iterative convergence history versus different LEAP2 buffer size of W-FETI and MG-FETI-LEAP2.

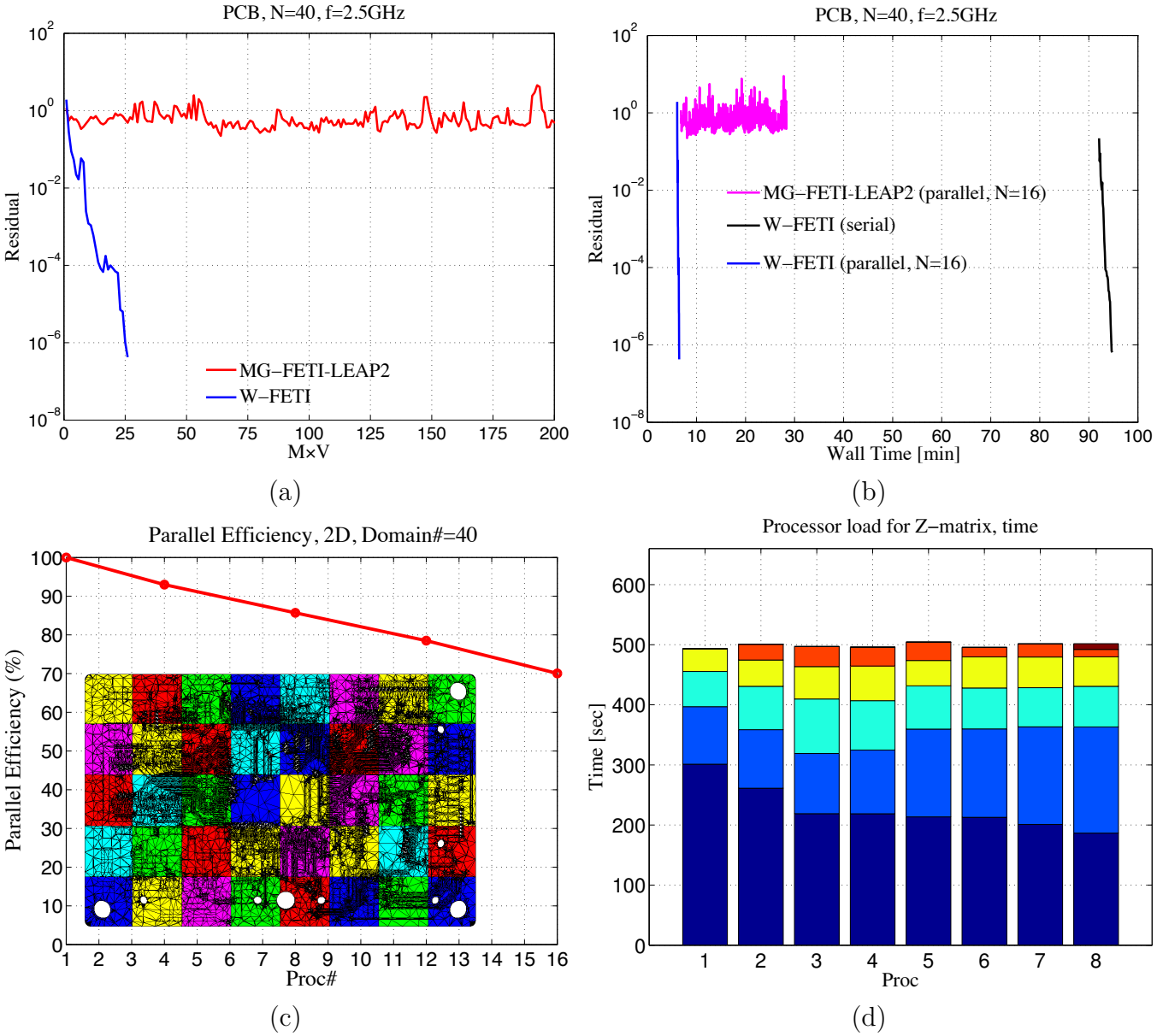


Figure 5.6. Iterative convergence and parallel efficiency the PCB SI analysis example decomposed into 40 domains: (a) Iterative convergence history comparison between W-FETI with $\varepsilon_{\text{svd}} = 0.01$ and MG-FETI; (b) Convergence history versus wall-time of W-FETI (serial and parallel) and MG-FETI-LEAP3; (c) Parallel efficiency versus number of CPU cores. (d) Computational time overhead for domain discrete DtN at each CPU core ($N_{core} = 8$).

Table 5.2. Computational statistics on a multi-layer PCB SI analysis example, minimum LEAP2 buffer used.

Domain count	Method	Unknown number (M)	MxV (tol=10^{-6})	Time time (hh:mm:ss)	Memory [GB]
15	W-FETI[a]	7.23	9	00:34:32[b]	8.64
	MG-FETI-DirectZ	7.23	–[c]	–	12.89
40	W-FETI	7.21	26	00:07:08[d]	8.32
	MG-FETI-DirectZ	7.21	–	–	–

[a]With compressed representation of domain discrete DtN oeprators using randomized algorithms.

[b]CPU core#=15

[c]Not able to converge.

[d]CPU core#=40

Table 5.3. Computational statistics on a multi-layer PCB SI analysis example with various LEAP2 buffer sizes (serial run, $N_D = 15$).

LEAP^2 Buffer	Method	MxV (tol=10^{-6})	Total Time (hh:mm:ss)	LEAP^2 Mem [MB]	Memory [GB]
minimum	MG-FETI-LEAP2	–[a]	–	57.34	9.084
	W-FETI	600	07:48:12	57.34	9.042
moderate	MG-FETI-LEAP2	1760	12:28:04	1,955.84	12.89
	W-FETI	520	07:05:29	1,955.84	12.42
maximum	MG-FETI-LEAP2	956	11:12:42	10,731.52	19.51
	W-FETI	346	07:44:59	10,731.52	19.11

[a]Not able to converge.

5.3 Signal Integrity Analysis of Integrated Circuit (IC) Package

The proposed methodology was used to perform the full-wave analysis of a complex integrated circuit (IC) package benchmark set by IBM. This benchmark is often termed the EPEP 2006 benchmark as it was set during a special session at the 15th Conference on Electrical Performance of Electronic Packaging by Gjonaj et. al. [79].

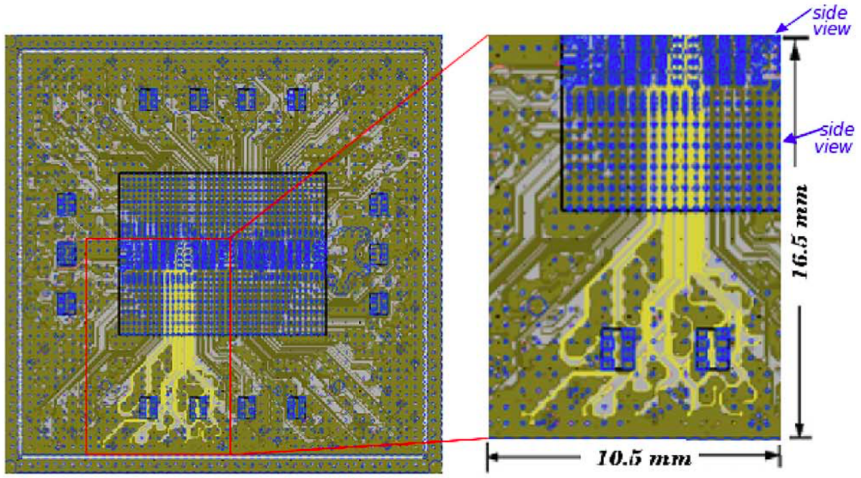
A portion of the IC package is finalized for the particular set of excited traces as shown in Fig. 5.7(a). In this work, domains partitioning was not optimized to lead to the minimal computational resources as discussed in [2], in order to examine how the proposed method performs in a worst case scenario, when expert user input is required. The IC package was discretized with 1,520,593 tetrahedra that lead to 8,275,688 second-order first-kind Nedelec TV-FEM unknowns and 238,268 dual unknowns. The smallest element corresponds to $\lambda/980$ at the frequency of operation and the average discretization size is $\lambda/90$. The model is is partitioned in a geometrical and mesh non-confirming manner into 51 domains in two dimensional fashion as shown in Fig. 5.7(b)-(c). This model uses thick conductors that are modeled as PECs and has six metal layers and more than 3,000 vias as shown in the mesh view of Fig. 5.7(d). The model is chosen to solve at 8 GHz since it was the most difficult frequency to converge for MG-FETI-LEAP2. A W-FETI SVD truncation tolerence was set to $\varepsilon_{\text{svd}} = 10^{-2}$. The placement of two waveports are shown in Fig. 5.8 whereas the induced electric currents are plotted in the same figure. The proposed W-FETI converges to a residual of 10^{-4} in 33 iterations, i.e. 66 matrix-vector-products whereas MG-FETI-LEAP2 needs more than 350 matrix-vector-products to converge to the same residual, as shown in Fig. 5.9(a). The convergence history of error residual versus run-time when a total number of 51 CPU cores are used is plotted in Fig. 5.9(b), with W-FETI and MG-FETI-LEAP2. This plot suggests that the proposed W-FETI reaches the desired error residual for about the half amount of

wall-time that the MG-FETI-LEAP2 takes. It is noted that an preconditioned FETI or FETI-DP method would not have converged at all in this problem. By employing the randomized discrete DtN computation with adaptive error tolerence `tol`= 0.1, the proposed W-FETI saves about 17% in memory and almost 50% in run-time compared to the uncompressed MG-FETI approach. A detailed description of the computational statistics are found in Table 5.4.

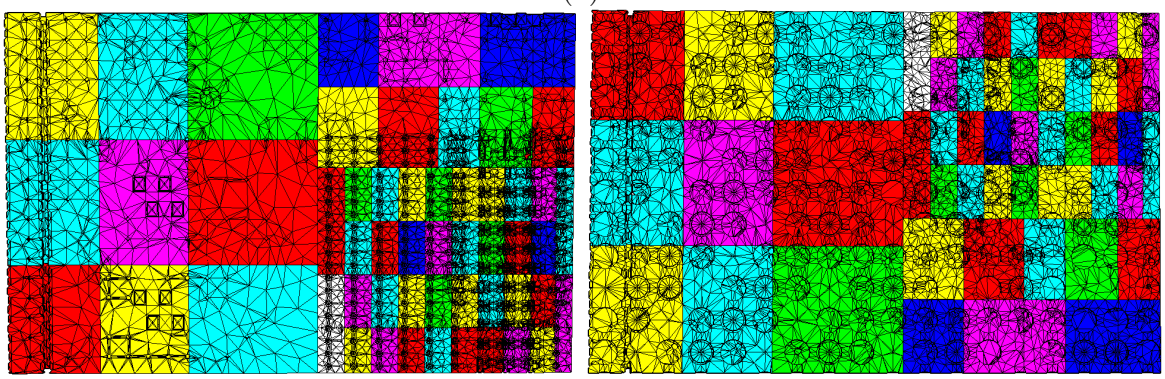
Table 5.4. Computational statistics on a multi-layer IC package analysis with 2D decomposition of 51 domains ($N_{\text{core}} = 51$).

Domain count	Method	Unknown number (M)	# of MxVs (tol=10^{-4})	Time (hh:mm:ss)	Memory [GB]
51	W-FETI[a]	8.51	66	00:14:27	28.17
	MG-FETI-LEAP2	8.51	366	00:28:37	34.1

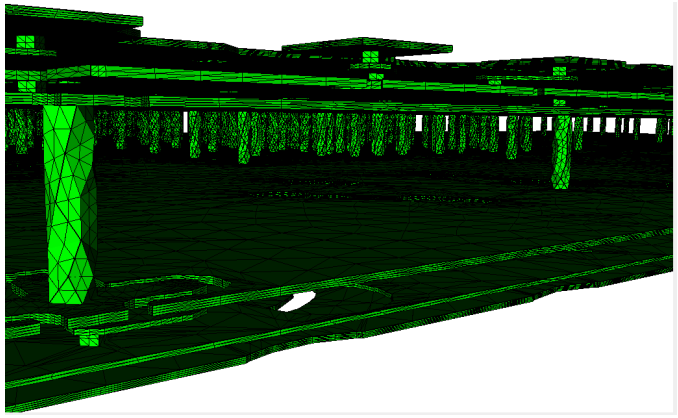
[a]With compressed representation of domain discrete DtN oeprators using randomized algorithms.



(a)



(b) (c)



(d)

Figure 5.7. Model overview, mesh and decomposition of the IC package example : (a) Overview of the IC package and the portion the simulation was performed on. [2]; (b) Front view; (c) Back view; (d) Side view.

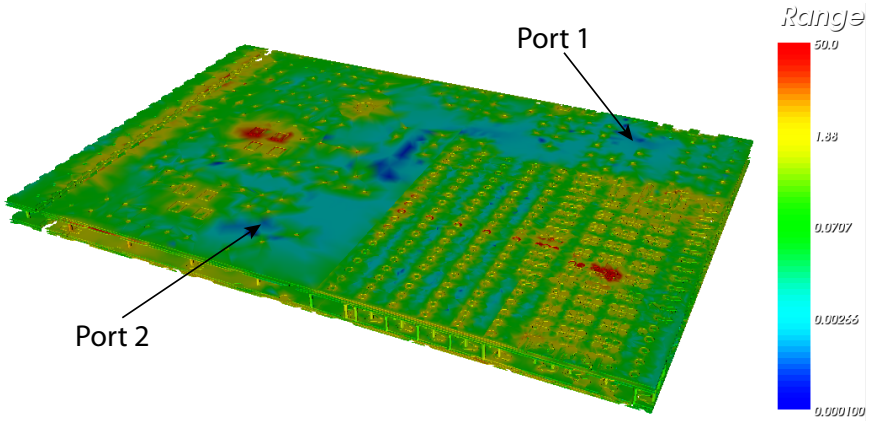
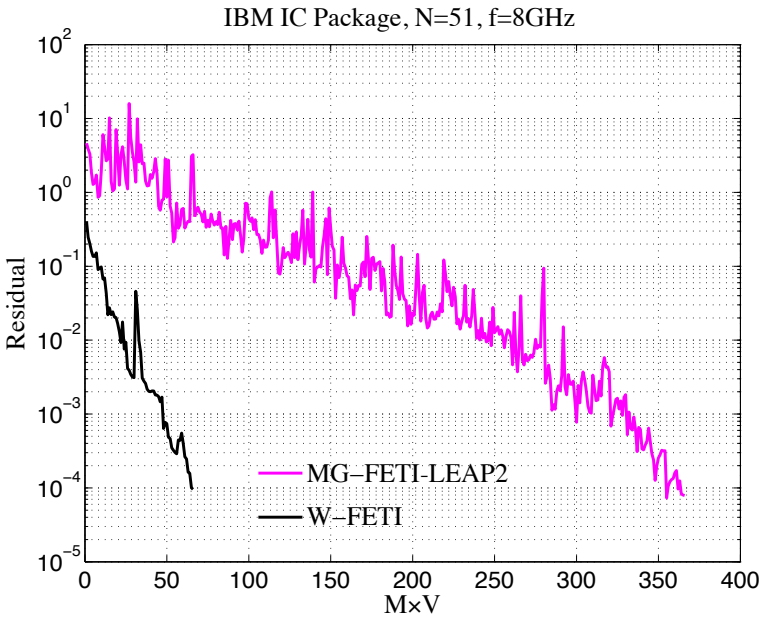
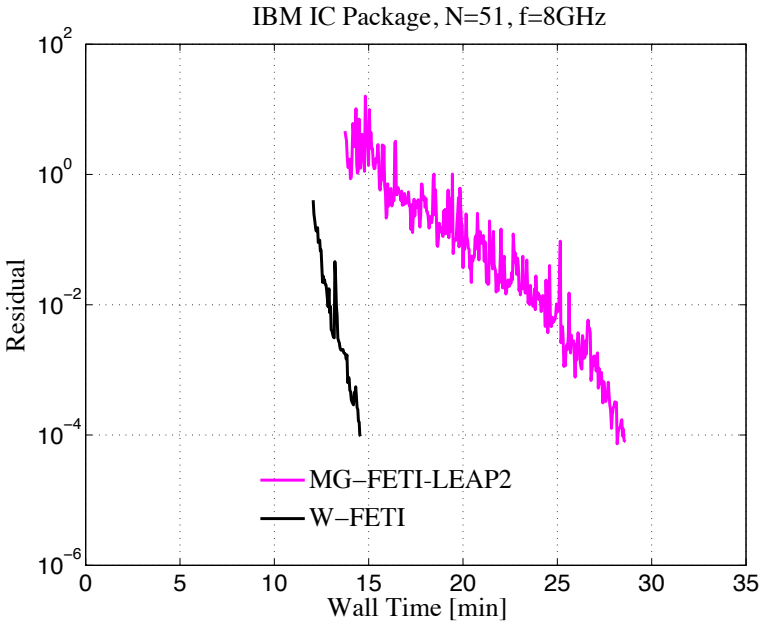


Figure 5.8. Induced electric currents on a multi-layer printed circuit board, $f = 8$ GHz.



(a)



(b)

Figure 5.9. Iteriative convergence and simultation time of the IBC IC Package model with 51 domains: (a) Iterative convergence history versus matrix-vector-products of W-FETI and MG-FETI-LEAP2; (b) Iterative convergence history versus wall-time of W-FETI and MG-FETI-LEAP2;.

5.4 Vivaldi Array Enclosed by A Radome

Enclosed by a radome, the $5\times4\times2$ dual-polarized 10:1 Vivaldi array [80] is considered next. Details on the geometry can be found in [80], this is a radiation problem that involves complicated antennas with large feature aspect radiation, and multi-layered dielectrics/radome. At the frequency of operation, f=18GHz, the element measured as $2.56 \times 0.5\lambda$ and the radome is measured as 7.4λ in diameter at its base and 3.8λ in height. The simulated array is detailed in Fig. 5.10. The model is discretized with 3,239,460 tetrahedrons, resulting a total number of 19,534,790 first-kind Nedelec second-order TV-FEM unknowns and 1,872,106 dual-unknowns. The problem is solved with 80 CPU cores with `tol`= 0.1, $\varepsilon_{\text{svd}} = 10^{-2}$ and iterative residual tolerence $\varepsilon = 10^{-4}$. The entire model is decomposed into 270 domains in a 3D topology, the Vivaldi array element and the radome model and their partitions are shown in Fig. 5.10(a)-(b), respectively, while the entire Vivaldi array without the radome is shown in Fig. 5.10(c). The radiated fields along the $\phi = 0^\circ$ cut at $f = 18$ GHz is plotted in Fig. 5.11. To compare the convergence performance with MG-FETI-LEAP3, Fig. 5.12(a) shows the error residual versus matrix-vector-products for both methods, showing W-FETI converges takes less than 40% MxVs than MG-FETI-LEAP3. With the randomized discrete DtN computation, the W-FETI approach saves about 18% in time compared to direct discrete DtN approach, as illustrated in the convergence history versus wall-time plot shown in Fig. 5.12(b). Computational details of the Vivaldi array simulation is given in Table 5.5.

Table 5.5. Computational statistics on a $5 \times 4 \times 2$ dual-polarized Vivaldi array enclosed in a multilayered radome ($N_{\text{core}} = 51$).

Domain count	Method	Unknown number (M)	MxV (tol=10^{-4})	Total time (h:m:s)	z-matrix memory [GB]	Total memory [GB]
270	W-FETI[a]	21.41	142	01:17:13	81.25	154.88
	MG-FETI-LEAP3	21.41	230	01:32:14	97.98	171.6

[a]With compressed representation of domain discrete DtN oeprators using randomized algorithms.

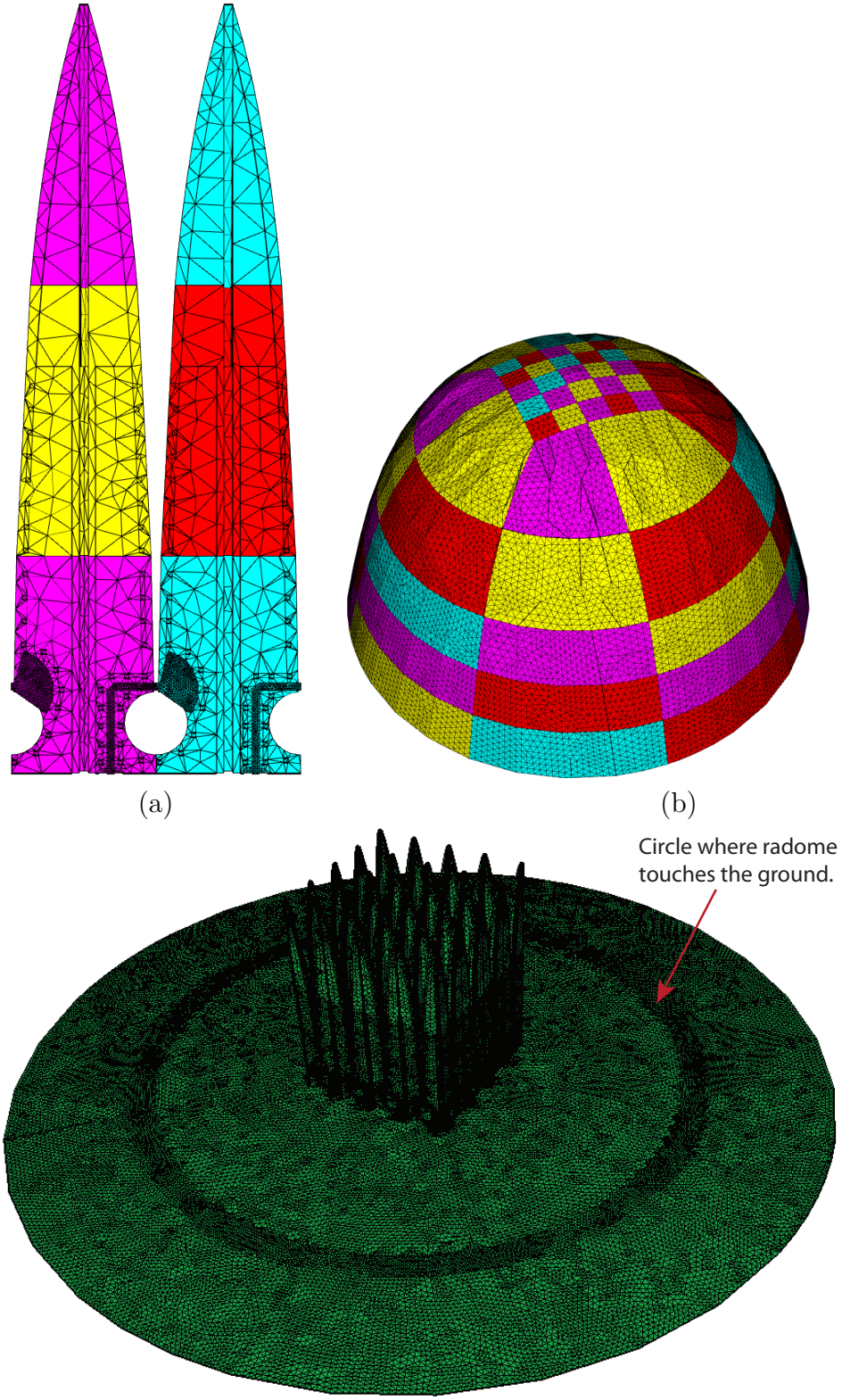


Figure 5.10. Computational model of the $5 \times 4 \times 2$ dual-polarized 10:1 Vivaldi array model enclosed by a multilayered radome: (a) Vivaldi element and domain partition; (b) Radome to enclose the Vivaldi array and its domain partition; (c) View of the entire Vivaldi array and ground plane without showing the radome.

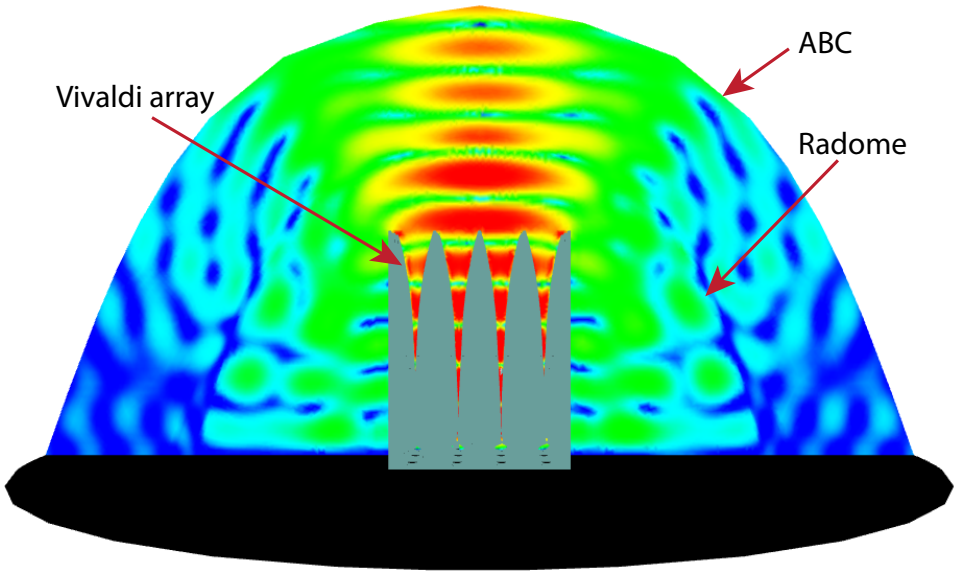
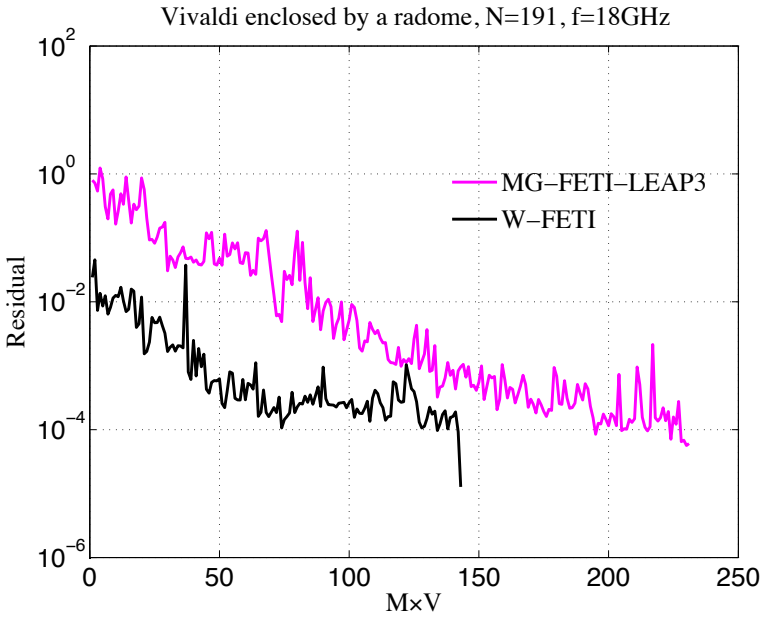
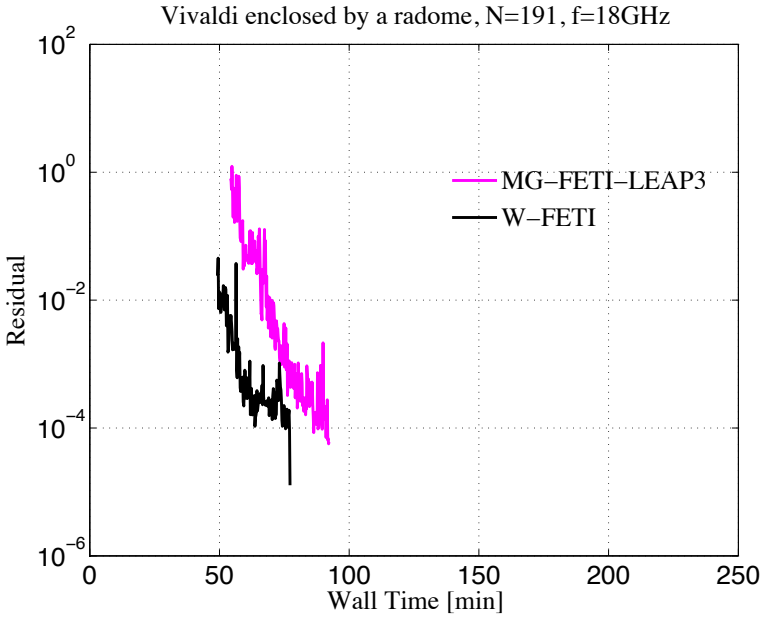


Figure 5.11. The radiated fields from the $5 \times 4 \times 2$ Vivaldi array enclosed in a multilayered radome, $f = 18$ GHz under broadside excitation.



(a)



(b)

Figure 5.12. Iterative convergence of a $5 \times 4 \times 2$ dual-polarized 10:1 Vivaldi array model enclosed by a multilayered radome ($N_{\text{core}} = 80$): (a) Iterative convergence history versus matrix-vector-products of W-FETI and MG-FETI-LEAP3; (b) Iterative convergence history versus wall-time of W-FETI and MG-FETI-LEAP3.

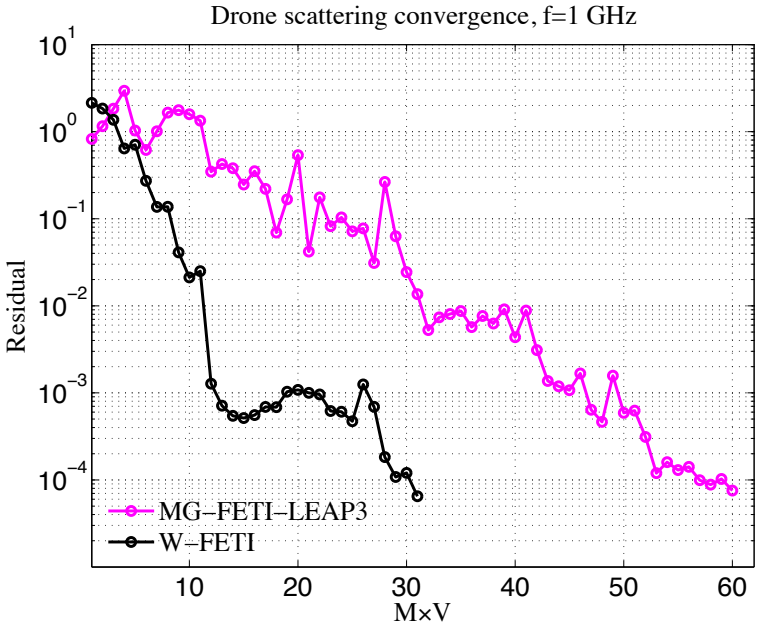
5.5 Scattering by A Generic Drone Aircraft

The last numerical example is the same challenging scattering problem presented in Chapter 3. The considered drone model is due to an oblique $\theta = \phi = 45^\circ$ incident planewave with E_x polarized. The entire geometry is decomposed into 432 domains with 40,350,720 first-kind Nedelec second-order TV-FEM unknowns and 3,439,672 dual-unknowns. The problem is solved with 88 CPU cores for both W-FETI and MG-FETI-LEAP3. A SVD truncation parameter $\varepsilon_{\text{svd}} = 10^{-2}$ is used for this experiment. The proposed W-FETI converges at $\varepsilon = 10^{-4}$ with 32 matrix-vector-products whereas MG-FETI-LEAP3 needs 60 matrix-vector-products, as shown in Fig. 5.13(a). Moreover, by combining the randomized discrete DtN computation, the proposed DD saves almost 20% in time and 28% in memory cost compared to its competitor. The convergence history of error residual versus run-time is plotted in Fig. 5.13(b). Detailed computational statistics regarding this simulation is given in Table 5.6.

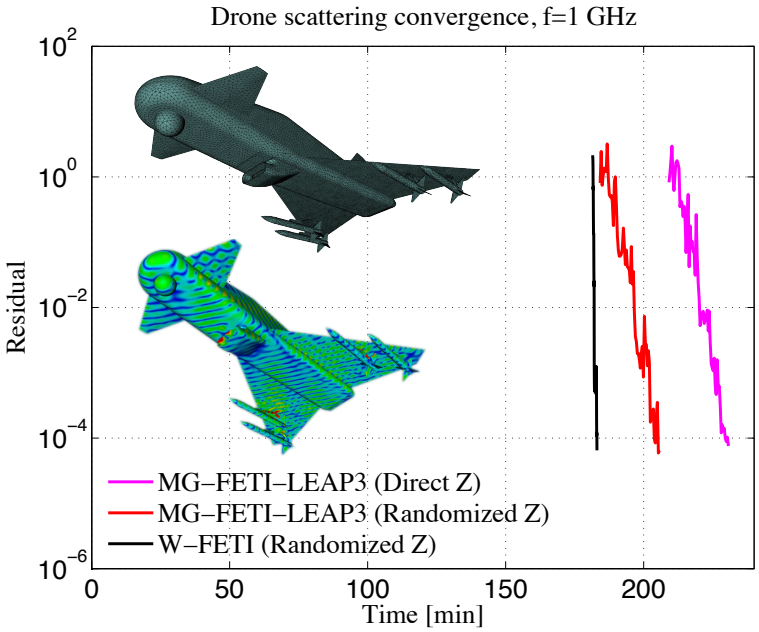
Table 5.6. Computational statistics on a generic drone aircraft model ($N_{\text{core}} = 88$).

Domain count	Method	Unknown number (M)	# of MxVs (tol=10^{-4})	Time (hh:mm:ss)	z-matrix Mem [GB]	Memory [GB]
432	W-FETI[a]	43.2	32	03:03:18	138.8	211.53
	MG-FETI-LEAP3	43.2	60	03:50:21	228	296.15

[a]With compressed representation of domain discrete DtN oeprators using randomized algorithms.



(a)



(b)

Figure 5.13. Iterative convergence of a generic drone aircraft model under an oblique $\theta = \phi = 45^\circ$ E_x polarized incident planewave at $f = 1$ GHz ($N_{\text{core}} = 88$): (a) Iterative convergence history versus matrix-vector-products of W-FETI and MG-FETI-LEAP3; (b) Iterative convergence history versus wall-time of W-FETI and MG-FETI-LEAP3.

CHAPTER 6

EPILOGUE

6.1 Summary

This dissertation introduces an Finite Element Tiering and Interconnecting (FETI) domain decomposition framework that leverages randomized linear algebra algorithms to significantly computational reliability, efficiency and scalability. Chapter 1 introduces the fudnamental issues in developing computational methods for multiscale EM modeling. A literature review of the state-of-the-art DD methods and randomized linear algebra computations is presented. Chapter 1 closes with a clear statement of the dissertation contributions.

Chapter 2 mainly introduces notations, conventions, mathematical spaces and offers a basic introduction to the FETI-2λ method that is used in the rest of the dissertation.

Chapter 3 and Chapter 4 present the core work of this dissertation. Chapter 3 starts with a fundamental observation that the off-diagonal FETI-2λ matrix blocks representing DoFs interaction residing on different interfaces are rank deficient. Following this observation, a simple and reliable method based on randomized rank-revealing algorithm is devised to efficiently compute and store the domain discrete DtN map operators. A detailed complexity and error study of the proposed approach is also provided. Finally, Two numerical examples are presented to illustrate the effectiveness and efficiency of the proposed method.

Chapter 4 also uses the low-rank matrix observation of FETI-2λ to construct a very effective, reliable and "black-box" global preconditioner. This global precondi-

tioning framework uses Woodbury matrix identity, thus termed as Woodbery FETI (W-FETI). Two important techniques, one based on a randomized SVD and one based on the sparsity of the FETI matrix are introduced to efficiently construct and apply the global preconditioner with the locally exact algebraic preconditioners (LEAP) proposed in [1] are also discussed. Chapter 4 closes with a detailed numerical study of the proposed W-FETI method on several simple examples.

Chapter 5 presents five realistic challenging problems to showcase the achieved accuracy, efficiency and robustness of the proposed DD framework for EM computation. Namely, a waveguide filter designed to operate at X-band, signal integrity analysis problem on one multilayered PCB and one IC package, a radiation problem of a $5 \times 4 \times 2$ ultra-wideband Vivaldi array enclosed by a radome and a scattering by a generic drone aircraft.

6.2 Conclusions

It was shown that randomized computations can be used to not only accelerate deterministic finite element domain decomposition computations in electromagnetics, but they can also improve the numerical resilience and parallel scalability of such tools.

The fundamental observation in integrating randomized computations with deterministic FEM based computations is the low-rank nature of the discrete DtN map operator of the FETI interface equations. Although this observation is not new in electromagnetics, this is the first attempt in FEM/DD computations. Specifically, a randomized and error controllable low-rank matrix decomposition method is used to accelerate the computation of domain discrete DtN map operator but also to construct a global preconditioner that is derived from the Woodbery matrix inversion identity, and is termed as W-FETI. The randomized discrete domain DtN solver works surprisingly well, resulting in 18%-40% total time and memory savings.

The proposed W-FETI preconditioner addresses an important challenge in developing scalable DD methods. The resulting global preconditioning matrix is controllable in size and can be efficiently assembled in a parallel computing environment. The proposed framework requires minimal user expertise in CEM other than using some error tolerance parameters 0.1 or 0.01 for more conservative choice. The developed preconditioner automatically allocates rescuers where they are due. Moreover, the Woodbury matrix identity based formula naturally brings the local preconditioners (LEAP^p) developed in [1]. Numerical results suggest that the proposed W-FETI-LEAP^p preconditioning approach is suitable for problems with structured and unstructured decomposition, non-conforming meshes across interfcaes, and it is scalable w.r.t. domain discretization, count and operating frequencies. Iterative convergence results show that W-FETI performs as good as the state-of-art for simple and one-way decomposition problems, takes 30%-82% less MxVs than the state-of-art for some real-life challenging 2D/3D decomposition problems.

In summary, the proposed work offers a promising methodology that significantly enhanced the efficiency, numerical resilience and parallel scalability of the state-of-the-art DDM electromagnetic computations. More importantly, proposed approach is a "black-box" algebraic one, therefore it can be readily extended to other computational disciplines such as acoustics, mechanics, fluid dynamics, or quantum mechanics.

BIBLIOGRAPHY

[1] G. N. Paraschos, *Robust and Scalable Domain Decomposition Methods for Electromagnetic Computations*. PhD dissertation, University of Massachusetts Amherst, Aug. 2012.

[2] Y. Shao, Z. Peng, and J.-F. Lee, "Full-wave real-life 3-d package signal integrity analysis using nonconformal domain decomposition method," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 59, no. 2, pp. 230–241, 2011.

[3] L. Hamandi, R. Lee, and F. Ozguner, "Review of domain-decomposition methods for the implementation of fem on massively parallel computers," *Antennas and Propagation Magazine, IEEE*, vol. 37, no. 1, pp. 93–98, 1995.

[4] S. Toledo, "A survey of out-of-core algorithms in numerical linear algebra," 1999.

[5] N. Halko, P. G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decomposition," *SIAM Review*, vol. 53, no. 2, pp. 217–288, 2011.

[6] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, "A fast randomized algorithm for the approximation of matrices," *Applied and Computational Harmonic Analysis*, vol. 25, no. 3, pp. 335 – 366, 2008.

[7] P.-G. Martinsson, V. Rokhlin, and M. Tygert, "A randomized algorithm for the decomposition of matrices," *Applied and Computational Harmonic Analysis*, vol. 30, no. 1, pp. 47–68, 2011.

[8] D. Fallis, "The reliability of randomized algorithms," *The British Journal for the Philosophy of Science*, vol. 51, no. 2, pp. 255–271, 2000.

[9] M. T. Goodrich and R. Tamassia, "Simplified analyses of randomized algorithms for searching, sorting, and selection," *work*, vol. 53, p. 57, 1999.

[10] J. Yang, X. Meng, and M. W. Mahoney, "Implementing randomized matrix algorithms in parallel and distributed environments," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 58–92, 2016.

[11] J. Song, C.-C. Lu, and W. C. Chew, "Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects," *IEEE Trans. Antennas Propag.*, vol. 45, no. 10, pp. 1488–1493, 1997.

[12] O. Ergul and L. Gurel, *The multilevel fast multipole algorithm (MLFMA) for solving large-scale computational electromagnetics problems*. John Wiley & Sons, 2014.

[13] J.-F. Lee and D.-K. Sun, "p-Type Multiplicative Schwarz (pMUS) Method with Vector Finite Elements for Modeling Three-Dimensional Waveguide Discontinuities," *IEEE Trans on Microw. Theory Tech.*, vol. 52, no. 3, pp. 864–870, 2004.

[14] K. Zhao, M. N. Vouvakis, and J.-F. Lee, "Solving electromagnetic problems using a novel symmetric fem-bem approach," *IEEE Transactions on Magnetics*, vol. 42, pp. 583–586, April 2006.

[15] M. N. Vouvakis, *A Non-Conformal Domain Decomposition Method for Solving Large Electromagnetic Wave Problems*. PhD dissertation, The Ohio State University, 2005.

[16] Z. Peng, X. C. Wang, and J. F. Lee, "Integral equation based domain decomposition method for solving electromagnetic wave scattering from non-penetrable objects," *IEEE Transactions on Antennas and Propagation*, vol. 59, pp. 3328–3338, Sept 2011.

[17] ANSYS, "HFSS." `http://www.ansys.com/Products/Simulation+Technology/Electronics/Signal+Integrity/ANSYS+HFSS`. Accessed: 12-05-2015.

[18] Altair, "FEKO." `https://www.feko.info`. Accessed: 12-05-2015.

[19] H. A. Schwarz, "II. Ueber einen Grenzübergang durch alternirendes Verfahren.." Wolf J. XV. 272-286. 1870 (1870)., 1870.

[20] A. Toselli and O. Widlund, *Domain Decomposition Methods - Algorithms and Theory*. Springer, 2005.

[21] J. Mandel, "Balancing domain decomposition," *Comm. in Numer. Methods in Engrg.*, vol. 9, no. 3, pp. 233–241, 1993.

[22] C. Farhat, A. Macedo, and M. Lesoinne, "A two-level domain decomposition method for the iterative solution of high frequency exterior Helmholtz problems," *Numer. Math.*, vol. 85, pp. 283–308, 2000.

[23] A. F. Peterson, *Computational Methods for Electromagnetics*. Wiley-IEEE Press, Dec. 1997.

[24] C. Dohrmann, "A preconditioner for substructuring based on constrained energy minimization," *SIAM Journal on Scientific Computing*, vol. 25, pp. 246–258, 2003.

[25] C. Farhat and F.-X. Roux, "A method of finite element tearing and interconnecting and its parallel solution algorithm," *Int. J. Numer. Math. Engng.*, vol. 32, no. 6, pp. 1205–1227, 1991.

[26] C. Wolfe, U. Navsariwala, and S. D. Gedney, "A parallel finite-element tearing and interconnecting algorithm for solution of the vector wave equation with pml absorbing medium," *IEEE Trans. Antennas Propag.*, vol. 48, pp. 278–284, 2000.

[27] M. N. Vouvakis, Z. Cendes, and J.-F. Lee, "A FEM Domain Decomposition Method for Photonic and Electromagnetic Band Gap Structures," *IEEE Trans. Antenna and Propag.*, vol. 54, pp. 721–733, Feb. 2006.

[28] Y. Li and J.-M. Jin, "A vector dual-primal finite element tearing and interconnecting method for solving 3-D large-scale electromagnetic problems," *IEEE Trans. Antennas Propag.*, vol. 54, no. 10, pp. 3000–3009, 2006.

[29] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen, "FETI-DP: a dual-primal unified FETI method - part I: A faster alternative to the two-level FETI method," *Int. J. Numer. Meth. Engng.*, vol. 50, no. 7, pp. 1523–1544, 2001.

[30] Y. Li and J.-M. Jin, "A new dual-primal domain decomposition approach for finite element simulation of 3-d large-scale electromagnetic problems," *IEEE Trans. Antennas Propag.*, vol. 55, pp. 2803–2810, 2007.

[31] B. Smith, P. Bjorstad, and W. Gropp, *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*. Cambridge university press, 2004.

[32] C. C. Stolk, "A rapidly converging domain decomposition method for the helmholtz equation," *Journal of computational Physics*, vol. 241, pp. 240–252, 2013.

[33] P.-L. Lions, "On the Schwarz Alternating Method III: A Variant for Nonoverlapping Subdomains," in *Third International Symposium on Domain Decomposition Methods*, (Philadelphia, PA), SIAM, 1989.

[34] B. Despres, P. Joly, and J. E. Roberts, "A domain decomposition method for the harmonic Maxwell equations in iterative methods in linear algebra," *Iterative Methods in Linear Algebra*, pp. 245–252, 1992.

[35] V. Rawat, *Finite element domain decomposition with second order transmission conditions for time-harmonic electromagnetic problems*. PhD dissertation, Ohio State University, 2009.

[36] C. Farhat, J. Mandel, and F. X. Roux, "Optimal convergence properties of the FETI domain decomposition method," *Computer Methods in Applied Mechanics and Engineering*, vol. 115, pp. 365–385, 1994.

[37] F. Magoules, F.-X. Roux, and S. Salmon, "Optimal discrete transmission conditions for a nonoverlapping domain decomposition method for the Helmholtz equation," *SIAM J. Sci. Comput.*, vol. 25, pp. 1497–1515, May 2004.

[38] M. D. B. Smith and O. Widlund, "Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions," *SIAM J. Numer. Anal.*, vol. 31, no. 6, pp. 1662–1694, 1994.

[39] A. Toselli and X. Vasseur, "Dual-primal FETI algorithms for edge element approximations: Two-dimensional h and p finite elements on shape-regular meshes," tech. rep., ETH Zurich, 2004.

[40] A. Toselli, "Dual-primal FETI algorithms for edge finite-element approximations in 3D," *IMA J. Numer. Anal.*, pp. 96–130, 2006.

[41] Z. Peng and J.-F. Lee, "A scalable nonoverlapping and nonconformal domain decomposition method for solving time-harmonic maxwell equations in $\mathbb{R}^3$," *SIAM J. Sci. Comput.*, vol. 34, pp. 1266–1295, May 2012.

[42] S.-C. Lee, M. N. Vouvakis, and J.-F. Lee, "A non-overlapping domain decomposition method with non-matching grids for modeling large finite antenna arrays," *Comput. Phys.*, vol. 203, no. 1, pp. 1–21, 2005.

[43] F. Ercal, J. Ramanujam, and P. Sadayappan, "Task allocation onto a hypercube by recursive mincut bipartitioning," in *Proceedings of the third conference on Hypercube concurrent computers and applications: Architecture, software, computer systems, and general issues-Volume 1*, pp. 210–221, ACM, 1988.

[44] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[45] Y.-J. Li and J.-M. Jin, "Parallel implementation of the feti-dpem algorithm for general 3d em simulations," *Journal of Computational Physics*, vol. 228, no. 9, pp. 3255–3267, 2009.

[46] K. Zhao, M. N. Vouvakis, and J.-F. Lee, "The adaptive cross approximation algorithm for accelerated method of moments computations of EMC problems," *IEEE Trans. Electromagnetic Compatibility*, vol. 47, no. 4, pp. 763–773, 2005.

[47] J. Tamayo, A. Heldring, and J. Rius, "Multilevel adaptive cross approximation (mlaca)," *Antennas and Propagation, IEEE Transactions on*, vol. 59, pp. 4600–4608, Dec 2011.

[48] D. Ludick, *Efficient numerical analysis of finite antenna arrays using domain decomposition methods*. PhD thesis, Stellenbosch University, 2014.

[49] K. Zhao, V. Rawat, and J.-F. Lee, "A domain decomposition method for electromagnetic radiation and scattering analysis of multi-target problems," *Antennas and Propagation, IEEE Transactions on*, vol. 56, no. 8, pp. 2211–2221, 2008.

[50] M. Bebendorf, "Approximation of boundary element matrices," *Numer. Math.*, vol. 86, no. 4, pp. 565–589, 2000.

[51] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American statistical association*, vol. 44, no. 247, pp. 335–341, 1949.

[52] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev, "Why the monte carlo method is so important today," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 6, pp. 386–392, 2014.

[53] M. W. Mahoney, "Randomized algorithms for matrices and data," *Foundations and Trends® in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.

[54] A. J. Kleiner, *Randomized Algorithms for Scalable Machine Learning*. PhD thesis, University of California, Berkeley, 2012.

[55] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[56] W.-B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," *Contemporary Mathematics*, vol. 26, pp. 189–206, 1984.

[57] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," in *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pp. 159–168, ACM, 1998.

[58] P.-G. Martinsson, V. Rockhlin, and M. Tygert, "A randomized algorithm for the approximation of matrices," tech. rep., DTIC Document, 2006.

[59] T. Sarlos, "Improved approximation algorithms for large matrices via random projections," in *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pp. 143–152, IEEE, 2006.

[60] Y. Zhu and A. Cangellaris, *Multigrid Finite Element Methods for Electromagnetic Field Modeling*. Wiely, 2006.

[61] M. A. Woodbury, *Inverting modified matrices*. Statistical Research Group, Memo. Rep. no. 42, Princeton University, Princeton, N. J., 1950.

[62] R. F. Harrington, *Time-Harmonic Electromagnetic Field*. Wiley-Interscience, 2001.

[63] D. M. Pozar, *Microwave Engineering*. John Wiley & Sons, 2005.

[64] S. J. A and L. J. Chu, "Diffraction theory of Electromagnetic Waves," *Physical Review*, vol. 56, pp. 99–107, July 1939.

[65] P. Sonneveld and M. B. van Gijzen, "IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations," *SIAM J. Sci. Comput.*, vol. 31, pp. 1035–1062, 2008.

[66] J. Nagar, G. Paraschos, and M. Vouvakis, "Robust and scalable finite element domain decomposition solvers for CEM," in *Antennas and Propagation Society International Symposium (APSURSI), 2013 IEEE*, pp. 1624–1625, July 2013.

[67] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM Journal of Matrix Analysis and Applications*, vol. 23, no. 1, pp. 15–41, 2001.

[68] W. Wang and M. N. Vouvakis, "Mesh morphing strategies for robust geometric parameter model reduction," in *Antennas and Propagation Society International Symposium (APSURSI), 2012 IEEE*, pp. 1–2, July 2012.

[69] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, pp. 3–30, Jan. 1998.

[70] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, "Basic linear algebra subprograms for fortran usage," *ACM Transactions on Mathematical Software (TOMS)*, vol. 5, no. 3, pp. 308–323, 1979.

[71] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, Y. Robert, F.-H. Rouet, and B. Uçar, "On computing inverse entries of a sparse matrix in an out-of-core environment," *SIAM journal on Scientific Computing*, vol. 34, no. 4, pp. A1975–A1999, 2012.

[72] Intel Math Kernel Library, Intel Corporation, https://software.intel.com/en-us/intel-mkl.

[73] M. A. Woodbury, "Inverting modified matrices," Tech. Rep. MR0038136, Princeton University, Princeton, NJ, USA, 1950.

[74] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.

[75] P. Monk, *Finite element methods for Maxwell's equations*. Oxford University Press, 2003.

[76] Open MPI, https://www.open-mpi.org.

[77] Bandpass Waveguide Filter with Dual Mode Cavities, CST, https://www.cst.com/Applications/Article/Bandpass-Waveguide-Filter-With-Dual-Mode-Cavities.

[78] G. L. Sleijpen and D. R. Fokkema, "Bicgstab (l) for linear equations involving unsymmetric matrices with complex spectrum," *Electronic Transactions on Numerical Analysis*, vol. 1, no. 11, p. 2000, 1993.

[79] B. Krauter, M. Beattie, D. Widiger, H.-M. Huang, J. Choi, and Y. Zhan, "Parallelized full package signal integrity analysis using spatially distributed 3d circuit models," in *Electrical Performance of Electronic Packaging, 2006 IEEE*, pp. 303–306, IEEE, 2006.

[80] D. Schaubert and T. Chio, "Wideband Vivaldi Arrays for Large Aperture Antennas," in *Proceedings of conference at the ASTRON Institute*, pp. 49–57, Apr. 1999.